NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DESIGN AND IMPLEMENTATION OF WIKI SERVICES IN A MULTILEVEL SECURE ENVIRONMENT

by

Kar Leong Ong

December 2007

Thesis Advisor:	Cynthia E. Irvine
Thesis Co-advisor:	Thuy D. Nguyen

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2007	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Design and Implementation of Wiki Services in a Multilevel Secure Environment		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Kar Leong Ong			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

The Monterey Security Architecture (MYSEA) provides a distributed multilevel secure networking environment where authenticated users can securely access data and services at different security classification levels. The MYSEA framework utilizes both commercial-off-the-shelf (COTS) products and specialized secure high assurance components that enforce multilevel security (MLS) policy. Some collaboration among MYSEA users is enabled through the use of the Web-based Distributed Authoring and Versioning (WebDAV) mechanism.

This thesis extends the existing collaboration capability in MYSEA to include hypertext content-based collaborative authoring and information sharing through the use of the increasingly popular wiki technology. This is accomplished by porting a publicly available wiki engine to run on a proprietary operating system hosting the MYSEA server. Through a systematic selection process, TWiki was chosen as the wiki engine for MYSEA. A three-stage porting methodology was used to aid in troubleshooting porting errors. Functional and security tests were performed to ensure that the wiki engine operates properly while being constrained by the underlying Mandatory Access Control (MAC) and Discretionary Access Control (DAC) enforcement mechanisms. This research is synergistic with the cross-domain information sharing emphasis fostered under various R&D programs in the DoD and intelligence communities.

14. SUBJECT TERMS Monterey Security Architecture, wiki, collaborative authoring, multilevel file sharing			**15. NUMBER OF PAGES** 142
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DESIGN AND IMPLEMENTATION OF WIKI SERVICES IN A MULTILEVEL SECURE ENVIRONMENT

Kar Leong Ong
Civilian, Ministry of Defense, Singapore
B.Eng., (Hons), National University of Singapore, 1997

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2007

Author: Kar Leong Ong

Approved by: Cynthia E. Irvine
Thesis Advisor

Thuy D. Nguyen
Co-Advisor

Peter J. Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Monterey Security Architecture (MYSEA) provides a distributed multilevel secure networking environment where authenticated users can securely access data and services at different security classification levels. The MYSEA framework utilizes both commercial-off-the-shelf (COTS) products and specialized secure high assurance components that enforce multilevel security (MLS) policy. Some collaboration among MYSEA users is enabled through the use of the Web-based Distributed Authoring and Versioning (WebDAV) mechanism.

This thesis extends the existing collaboration capability in MYSEA to include hypertext content-based collaborative authoring and information sharing through the use of the increasingly popular wiki technology. This is accomplished by porting a publicly available wiki engine to run on a proprietary operating system hosting the MYSEA server. Through a systematic selection process, TWiki was chosen as the wiki engine for MYSEA. A three-stage porting methodology was used to aid in troubleshooting porting errors. Functional and security tests were performed to ensure that the wiki engine operates properly while being constrained by the underlying Mandatory Access Control (MAC) and Discretionary Access Control (DAC) enforcement mechanisms. This research is synergistic with the cross-domain information sharing emphasis fostered under various R&D programs in the DoD and intelligence communities.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to express my sincere thanks to many individual who have assisted me throughout the course of this thesis study.

I would like to thank Prof. Cynthia Irvine and Thuy Nguyen for the time and effort in helping me to formulate the scope of the thesis, and guiding me throughout the length of my thesis research. Thank you for your patience and constructive comments in making the writing of this thesis paper a success.

I would like to thank Jean Khosalim and David Shifflett for their technical expertise and assistance in helping to troubleshoot the problems encountered during the implementation process. I am grateful to Jean for his patience and time in going through the installation instructions and test procedures to make them error free.

I would like to thank my sponsor, the Singapore Defense Science & Technology Agency, in giving me the opportunity to pursue this postgraduate study at the Naval Postgraduate School in Monterey.

Finally, I would like to share the success of the completion of this thesis research with my wife, who is constantly encouraging and supporting me.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

The use of wiki systems in corporate environments has become increasingly common. The ease of interaction and general operation for typical users makes wiki system an excellent tool for mass collaborative authoring. However, in a secure environment where users are subjected to strict access control policies for data with different security classifications, i.e., a multilevel security (MLS) environment, extra care should be taken to ensure that the wiki solutions are usable while still conforming to security policy. The motivation of this thesis is to demonstrate the feasibility of implementing a wiki solution in one such environment, i.e., the Monterey Security Architecture (MYSEA) [1].

B. PURPOSE

The purpose of this study is to determine whether it is possible to develop an architecture for incorporating wiki technology in a MLS environment and if so, to implement a wiki server in the MYSEA MLS Testbed. This study will enable collaborative authoring and sharing of information in a multilevel secure environment. Such a capability is beneficial in a coalition environment, whereby information with different sensitivities is shared among different entities

C. ORGANIZATION OF PAPER

This work is organized as follows:

- Chapter I provides the motivation and purpose of this study.
- Chapter II gives background information regarding wiki technology and the MYSEA project.
- Chapter III describes the selection process for candidate wiki engines.

- Chapter IV provides the project description, which includes the concept of operation and the methodology adopted for the study.
- Chapter V describes the wiki design and architectures, the problems encountered and the proposed solution.
- Chapter VI provides the testing procedure for the wiki implementation, and the results of the testing.
- Chapter VII concludes with a project summary, a discussion of related work, and suggestions for future work.

II. BACKGROUND

This chapter provides background information on various topics relevant to the implementation of a wiki in the MYSEA environment. The first part of the chapter gives an overview of wiki technology, and the second part covers the MYSEA project.

A. WIKI

Computer and communication technology have enabled the use of computer-supported cooperative work [2]. Such computer-supported collaboration technology provides a means for mass collaboration in which a group of people creates content collectively rather than individually, and one such mass collaboration tool is the wiki [3]. A wiki is a website that allows a visitor to view, add, edit and delete the content of the website. Such ease of interaction and operation makes a wiki an effective mass collaborative authoring tool. The wiki was first developed by Ward Cunningham in 1994, who named his work "wiki", a Hawaiian word for fast, which is meant to represent the working principle of such a mass collaborative authoring tool [4].

A wiki engine is usually implemented as server-side scripting technology [5], though there are also client-side technologies, but they are less commonly used. The wiki engine manages a set of documents known as wiki pages. A wiki page is written in plain text and stored either in a regular file system or in a database. When a browser requests a page, the wiki scripts translate the wiki page into HTML and returns it to the browser.

Besides the content, the returned HTML page also contains a header with links to scripts with specific functionalities. The most important of these links for a wiki is the "edit" link, which differentiates the wiki page from normal read-only webpages. When the "edit" link is clicked, the script returns the same wiki page, but in this case it is enclosed in a text-area form with a "save" link below. The reader can now edit the text and submit the new version via the "save" link, which immediately replaces the old version on the website. When the "save" link is clicked, the data in the text-area form is

sent to the wiki script, which stores the new text as a new version of the wiki page. The wiki engine usually controls the versioning of the wiki pages using a revision control system, and maintains a log of recent changes called a "history", thus allowing users to revert to previous versions of the wiki page.

The source format of the wiki page is augmented with a simplified non-standardized markup language to indicate various structural and visual conventions. Each wiki engine implements its own simplified markup language and there is little standardization among the wiki engines. HTML is not used because the tags make the actual text content hard to read. Therefore, plain text editing with a few simple conventions for structure and style is preferred in most wiki engine implementations. Hence, users only need to remember those few simple conventions and do not need to learn a cryptic code language like HTML. Another important feature of wiki technology is the wiki page link syntax, which is the syntax for creating hypertext links to other wiki pages [5]. Most wiki engines follow the syntax of enclosing a word or phrase in a [[double bracket]] to indicate a link to a wiki page with that name. Creating wiki page links is also the method for creating new pages. If the word or phrase enclosed in brackets is not the title of an existing page, a link is created that leads to the edit form for a new page having that title.

The wiki design outlined above permits easy creation and editing of page contents. It is the use of simplicity as an overarching design principle that makes wikis a widely used tool for mass collaborative authoring.

B. MYSEA

The Monterey Security Architecture (MYSEA) project aims at developing a secure architecture to enable the aggregation of data and services at different security classification levels into a distributed multilevel secure network. It is accomplished through the use of commercial off the shelf (COTS) products together with secure high assurance components that enforce multilevel security (MLS) policy. The high assurance products ensure that security is enforced across various sensitivity levels without having

physically separated hardware for networks at each sensitivity level. MYSEA aims to achieve a higher adoption rate than other MLS systems for organizations which already own a substantial number of COTS resources and do not wish to invest in new systems, by using a mix of COTS and special purpose components [1][6][7].

The multilevel security policy that the special purpose high assurance products enforce is formalized based on Bell-LaPadula mathematical model [8], where a subject is allowed to read an object only if the security level of the subject dominates that of the object, and write an object only if the security of the object dominates that of the subject. The products also enforce the integrity policy reflected in the Biba mathematical model [9], where a subject is allowed to read an object only if the integrity level of the object dominates that of the subject, and to write to an object only if the integrity level of the subject dominates that of the object.

Figure 1 shows the MYSEA network architecture. Communications between the clients on the MLS LAN and the services within the MLS enclave as well as the services on other connected single-level networks are handled via the XTS-400 high assurance MLS server. Once the session level is negotiated, a client can communicate as permitted by the enforced confidentiality and integrity policies. A Trusted Path Extension (TPE) device between the client and the XTS-400 server provides a mechanism for user identification, authentication and authorization, and supports negotiation and establishment of the session level.

The MYSEA MLS server supports HTTP, IMAP and SMTP protocols. The support of HTTP allows a wide range of applications such as the Tarantella web enablement software and WebDAV to be used. In particular, WebDAV provides the user with the ability to remotely access and modify files in the MLS context. A wiki, on the other hand, provides a means for rapid and distributed authoring of shared content. The goal of this thesis is to provide such mass collaborative authoring functionality through the implementation of wiki technology on the high assurance MYSEA server.

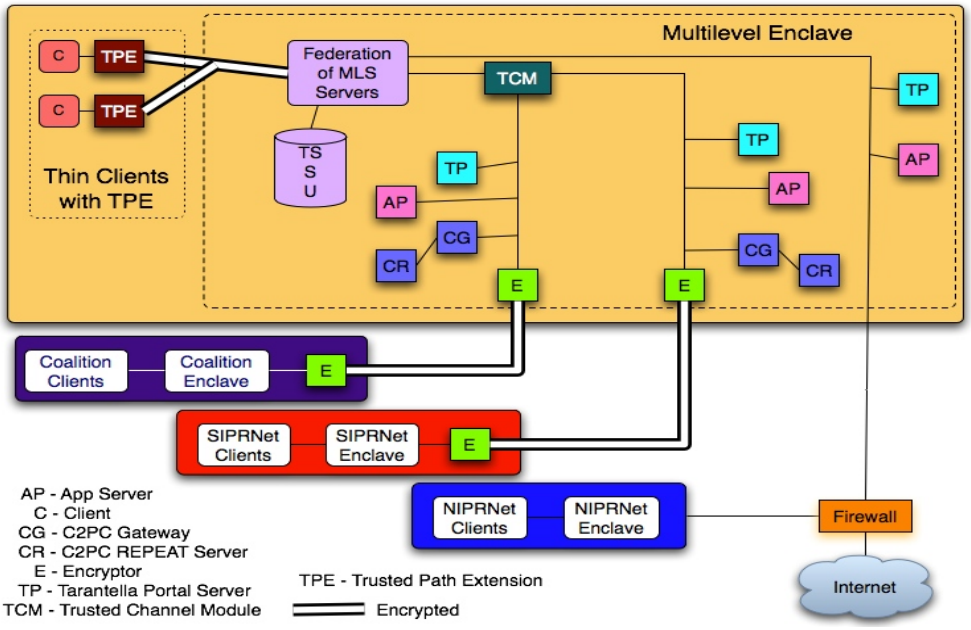

Figure 1. MYSEA Testbed Topology [From 10].

C. XTS-400 SERVER

BAE System's XTS-400 server is based on Intel x86 hardware and it runs the STOP operating system. STOP provides a Linux-like user and programming interface which allows many existing Linux applications to be ported to the XTS platform without significant modification. The XTS-400 server enforces both discretionary access control (DAC) and mandatory access control (MAC) security policies. Many features are available on the XTS-400 server, two are relevant to this study and are briefly described in the following sections.

1. Deflection Directory

In multilevel mode, STOP allows users to login to the XTS-400 at different session levels. They can read data with security labels at or below the established session level, but only write to data with security labels equal to the session level. If a directory is created at a particular level, the user will be able to take actions that would result in

changes to the directory only when logged in at the session level equal to that of the directory. Many typical applications use standard directory paths. To allow the use of the same apparent directory path at different session levels, STOP implements deflection directories. A deflection directory is a special directory that can be instantiated at different security levels simultaneously. When a user accesses a deflection directory, the OS automatically selects or creates an instance of the directory that corresponds to the user's current session level. A typical use of deflection directory is "*/tmp*". No data is shared between different instances of the deflection directories, and it is not possible to access instances of the directory at other levels. Although the use of deflection directories allows data to be written to what appears to be the same directory at different levels, the OS ensures that data within such directories is properly separated according to their security labels.

2. DAC Organization

In MYSEA, the Apache processes are instantiated with the userID of the authenticated user, i.e., the IDs of the Apache processes are the same as the ID of the logged in user. As such, any files created by the Apache process running on behalf of an user would be owned by that user. This is different from the standard Linux environment, where the Apache processes take on a dedicated userID, and any files created by those processes would be owned by that userID. This difference results in some challenges for this study, which will be elaborated in Chapter V.

D. SUMMARY

This chapter provided an overview of wiki technology, outlined background information on the MYSEA environment and discussed XTS-400's features of deflection directory and DAC organization. The next chapter will elaborate in details the selection criteria and outcome of the selection process used to choose an appropriate wiki engine for the MYSEA environment.

THIS PAGE INTENTIONALLY LEFT BLANK

III. WIKI ENGINE SELECTION

There are many different wiki engine implementations available, each of them differing in many aspects. To help in identifying a suitable wiki engine for use in this study, a structured approach in the selection process is adopted. This chapter outlines the wiki selection methodology, the selection criteria for the wiki engine for the MYSEA testbed, and the outcome of the selection process.

A. AVAILABLE WIKI ENGINES

Many different software technologies and solutions have been adopted in the open source community to implement various wiki engines. A search on wiki-related websites on the Internet returned about 140 different wiki engines[11], implemented using a wide range of programming languages such as C, Java, PHP, Perl, etc. Each of these wiki engines offers a differing breadth of functionalities. It is therefore crucial to adopt a sound methodology to aid in selection of a suitable wiki engine for use in the MYSEA testbed environment.

B. SELECTION METHODOLOGY

Due to the availability of many different wiki engines and the widespread adoption of wiki technology, many comprehensive comparisons of wiki engines have already been done and the information is available in the public domain. The approach taken here is therefore not to reinvent the wheel, i.e., to repeat those comparison, but to leverage existing information to help in the short listing process [12][13][14]. While reviewing the information, weights have been given to considerations of specific interest to the MYSEA testbed so as to make the selection relevant to the operating environment.

C. SELECTION CRITERIA

The primary consideration in selecting the wiki engine is that the wiki engine must be able to integrate into the MYSEA environment. The wiki engine must therefore be able to run on the RedHat 8.0 operating system, and be able to integrate with Apache HTTP Server Version 1.3.34. To keep the setup simple and to aid in troubleshooting, it is desirable that the selected wiki engines depend upon the standard file system instead of on special purpose databases for their data stores.

The maintainability of the software is another important consideration. A well maintained wiki engine will ensure that the wiki engine is up-to-date, and that public domain support is available should there be any issue. The Top-10 wiki engine list was used as a proxy measurement for the maintainability of the wiki engine. The rationale was that a more popular engine would imply that more people are using it and hence more people are willing to enhance and support it.

The supported functionality of the wiki engine is also an important consideration. The selected wiki engine should be able to support most of the essential features of a wiki engine. To aid in the comparative analysis, a popular wiki engine, MediaWiki, was used as the baseline to which the functionality of the wiki engine was compared. MediaWiki was chosen as the baseline because it is the underlying wiki engine for one of the most referenced wiki websites: Wikipedia.

D. SELECTION PROCESS

Upon application of the considerations outlined in the preceding paragraphs and using publicly available information [12][13], the list of wiki engines was narrowed from the initial list of more than 140 to a final list of two. Table 1 presents the detailed selection decisions and the elimination process. The two candidates were PmWiki and TWiki. PmWiki is a server-side wiki engine with an implementation based on the PHP

scripting language, and was started by Patrick R. Michaud [20]. TWiki is another server-side wiki engine started by Peter Thoeny, and is implemented using the Perl scripting language [21].

In order to select the final wiki, the following additional criteria were used: concurrent editing features, executable code size, process memory footprint, directory structure, and functionality design. The following paragraphs outline these criteria in greater detail.

1. Concurrent Edit Feature

As the main feature of a wiki is the ability to simultaneously edit the contents of a page by different users, it is therefore important that the wiki engine be able to handle this concurrent editing functionality. Specifically, the wiki engine must be able to handle the following scenario:

> Alice starts to edit a page. Before Alice saves her edits, Bob requests an edit of the same page, and receives the page's text prior to Alice's edits. Bob finishes with his edits and enters "save". Alice finishes editing her page, enters "save", and since she is working from a version of the page obtained before Bob has made his changes, she may wipe out Bob's edits in the process.

s/n	Wiki Engines	Elimination Process →							Shortlisted
		Regular FileSystem [1]	Top 10 Wiki List [2]	Interface with Apache [3]	Essential Functionality [3]	ACL [3]	Free & OpenSource [3]	Add-on functionality [3]	
1	KwikiKwiki	☑	☒	--	--	--	--	--	☒
2	DokuWiki	☑	☒	--	--	--	--	--	☒
3	JSPWiki	☑	☑	☒	--	--	--	--	☒
4	PerSpective	☑	☑	☒	--	--	--	--	☒
5	FlexWiki	☑	☑	☒	--	--	--	--	☒
6	UseModWiki	☑	☑	☑	☑	☒	--	--	☒
7	OddMuseWiki	☑	☑	☑	☑	☒	--	--	☒
8	TeleparkWiki	☑	☑	☑	☑	☑	☒	--	☒
9	MoinMoin	☑	☑	☑	☑	☑	☑	☒	☒
10	PmWiki	☑	☑	☑	☑	☑	☑	☑	☑
11	TWiki	☑	☑	☑	☑	☑	☑	☑	☑

Information Source: 1: http://c2.com/cgi-bin/wiki?WikiEngines
2: http://c2.com/cgi-bin/wiki?TopTenWikiEngines
3: http://www.wikimatrix.org

Table 1. Selection Decisions and Elimination Process.

PmWiki handles such a scenario by implementing a concurrent edit feature using the "diff3" UNIX command. Essentially, instead of saving Alice's edit, PmWiki presents a message saying someone else has changed the text while she is editing it. Bob's changes are merged into Alice's text, with any conflicts highlighted by parentheses. Alice can then fix the text as appropriate and save the updated pages. However, PmWiki does not inform Bob that someone is editing the page when he first requested to edit that page.

TWiki implements a similar concurrent edit feature using a Perl implementation of the "diff" command to show the difference between the versions. In addition, TWiki prompts Bob in advance, and gives him a choice of canceling or proceeding with the edit. TWiki is also able to handle the case whereby both users concurrently save the edits. PmWiki, on the other hand, handles this situation poorly, as one version of the edit will overwrite the other without invoking the concurrent edit warning prompt. In this aspect, TWiki is preferred over PmWiki.

2. Executable Code Size

Both PmWiki and TWiki are implemented using scripting languages. A comparison on the total executable code sizes of the scripts was obtained to serve as an indicator of the underlying complexity of the code.

PmWiki organizes its code into a main PHP script file called pmwiki.php, and supporting PHP scripts in the "script" directory. Users invoke PmWiki via the main PHP file, which then calls the other supporting scripts as necessary. The total size of the main file and 33 supporting scripts is 841Kbytes.

TWiki organizes its code into two directories: the "bin" directory for the main Perl scripts, and the "lib" directory for supporting Perl modules. User invokes TWiki via the "view" scripts in the "bin" directory, which then activates other supporting modules as necessary. The total size of the 230 script files in the "bin" and "lib" directories is 2262Kbytes.

In terms of code size, TWiki is more than two times the size of PmWiki. This provides an indication of the relative complexity of the two wiki engines. In this aspect, PmWiki is preferred over TWiki. However, the simplicity of PmWiki does have a downside in terms of functional robustness, which will be discussed in a later section.

3. Process Memory Footprint

The preceding paragraphs illustrate the relative complexity of PmWiki and TWiki through comparison of the executable code sizes. To further explore the process footprint of the two wiki engines during execution, their memory usages were monitored and compared.

The "top" command in Linux was used to list the amount of memory used by the two wiki engines. The "size" variable given by the "top" command gives the size of the process's code plus its data and stack space. In PmWiki, since PHP is compiled into the Apache code, the "top" command was used to monitor only the "httpd" process. It was found that PmWiki consumed about 4Mbytes of memory during execution. In TWiki, the Perl script interpreter is not integrated with the web server. Hence, there is a need to monitor both processes. It was found that the "httpd" process used 1016 Kbytes and the TWiki "view" script used between 5 and 12 Mbytes of memory, depending on the pages being requested.

In terms of memory footprint, TWiki occupies more memory compared to PmWiki. However, PmWiki process is continuously resident in memory since it is integrated with Apache server, while TWiki processs is non-resident as the Perl process is only invoked upon a page request. In this respect, the results are un-conclusive as the memory footprints of both wikis are not grossly disproportionate.

4. Directory Structure

Besides memory usage, another important aspect of wikis that must be considered in the MLS environment is the directory structure of each wiki engine. As the MLS environment enforces a mandatory access control policy, the directory and file structure

of the two candidate wiki engines, which are designed for use in a discretionary access control environment, must be able to accommodate such access control policy enforcement.

In PmWiki, the wiki pages are stored in the “wiki.d” directory. PmWiki only supports one level of sub-directory within the “wiki.d” directory. It does not support a hierarchical file structure within the “wiki.d” directory, but is able to support logical hierarchical groupings by naming the files according to a structured file naming convention that embeds the hierarchical information in the filename itself. The directory structure of PmWiki is illustrated in Figure 2 (the texts after the # symbol are comments and not part of the file or directory names).

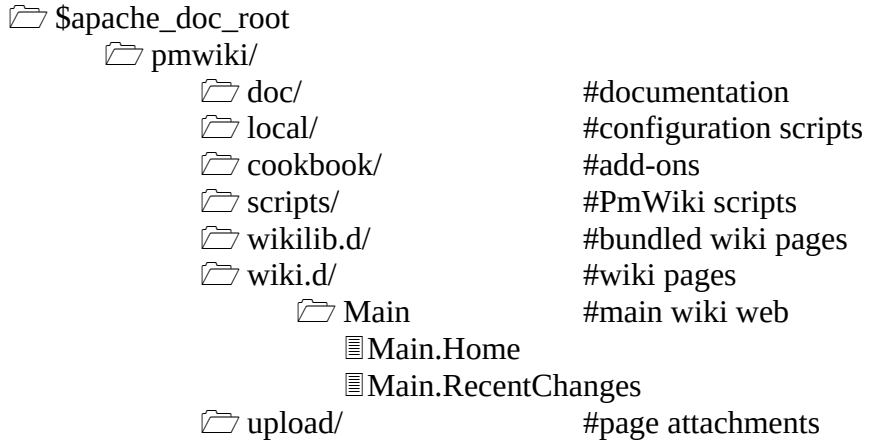

Figure 2. PmWiki Directory Structure.

In TWiki, the wiki pages are stored in the “data” directory. TWiki supports multiple levels of sub-directories within the “data” directory. Its logical hierarchical grouping is implemented using a corresponding hierarchical directory structure in the file system. The directory structure of TWiki is shown in Figure 3.

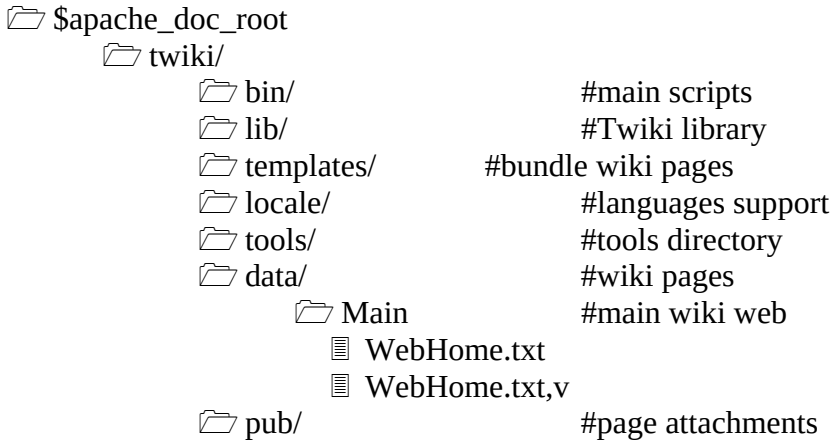

Figure 3. TWiki Directory Structure.

In terms of organization of the wiki pages to accommodate the mandatory policy, there are two possible approaches. The first is to organize the wiki pages by topic. In this approach, wiki pages are grouped by topics, which may consist of contents having different classifications. Figure 4 illustrates this approach.

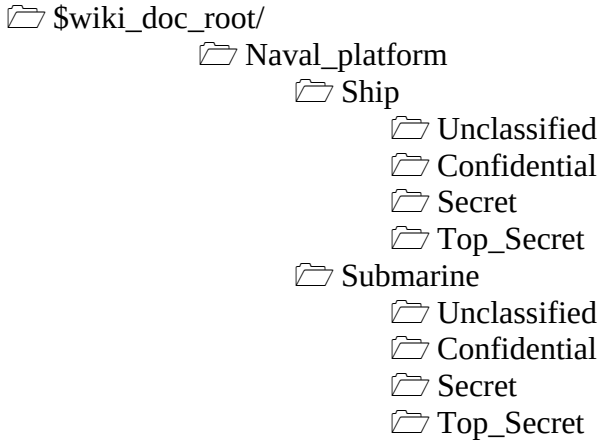

Figure 4. Organization of Wiki Pages by Topic.

The second approach is to organize the wiki pages by classification. In this case, wiki pages are first grouped by classification. Within each classification, the wiki pages

are then grouped into different topics. Each topic may appear in one or more classifications. Figure 5 shows an example of this approach.

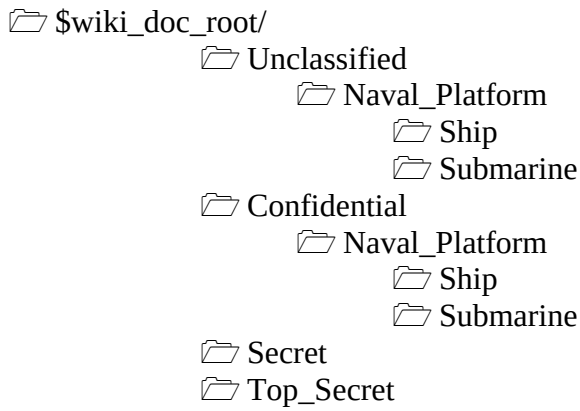

Figure 5. Organization of Wiki Pages by Classification.

Because TWiki supports a hierarchical directory structure in the "data" directory, it is able to support both approaches. PmWiki, on the other hand, can only support the organization of wiki pages by classification due to the fact that it only has one level of sub-directories within the "wiki.d" directory. Furthermore, all the PmWiki pages within each classification sub-directory will be stored in a flat file structure, differentiated only by the filename convention. This is undesirable because STOP OS limits the length of filenames to 256 characters, which means that there is a limit on the number of sub-directories supported. Furthermore, extremely long filenames complicate the task of file administration and housekeeping, since searching and locating such files in a text mode oriented user interface is very ineffective.

In the existing MYSEA testbed environment, the files within the Apache web server and the WebDav server are organized by classification. Therefore, the limitations of PmWiki do not pose a critical constraint, as the implementation of the wiki engine within the testbed environment would be based on a similar file system structure. However, TWiki does provide the option for a more intuitive organization of the wiki content, i.e., subdirectories within a classification. In this aspect, TWiki is preferred over PmWiki.

5. Functionalities Design

In a preceding section, it was shown that PmWiki has a smaller code size compared to TWiki. This is in part due to the fact that PmWiki's design philosophy is based on one of simplicity. As such, many functionalities are not included in the default installation package, and have to be enabled through additional configuration or installation of additional plug-ins. PmWiki and TWiki also differ in the way they implement access controls. One of the notable differences is that PmWiki allows multiple logins within the same browser session. This may occur when user attempts to access a page that his/her current login has no access rights. In such scenario, PmWiki will display a login prompt instead of an error message, at which point the user is allowed to access the page with another authorized account. However, the login names are not displayed anywhere within the wiki page to indicate the current active login. This could create confusion regarding the session status, especially if a user holds multiple login accounts. However, the most notable difference between the two wiki engines' access control implementation is in the history logging mechanism which keeps track of the audit log of all the previous changes in the wiki content. PmWiki allows users to arbitrarily specify the author name during the editing session. The history log will then display the specified author name, even if this is different from the login name. Hence, accountability is not being enforced in PmWiki's implementation. TWiki, on the other hand, enforces one login session per browser. Login sessions are distinguished through the display of the login ID on every page request, and the history log entries are bound to the identity of the login ID. In this respect, TWiki is considered to be more superior than PmWiki.

E. SELECTION OUTCOME

Table 2 summarizes the outcome of the selection process between PmWiki and TWiki outlined in the preceding paragraphs. Though the results showed that PmWiki has a smaller code size and memory footprint, such benefits are offset by the relatively inferior way it implements the wiki security-related functionalities such as system login

and history logging. Using the selection process, TWiki is deemed to be the more suitable choice for implementation in the MYSEA testbed environment.

S/N	Description	PmWiki	TWiki
1	Concurrent editing	Supported.	Supported. More robust implementation.
2	Size of executable code	841 KB	2250 KB
3	Footprint of process memory	4 MB	6 to 13 MB
4	Directory structure	Flat. Supports the organization of wiki pages by classification only.	Hierarchical. Supports both the organization of wiki pages by topic and classification.
5	Functionalities design	Accountability not enforced in access control.	Accountability is strictly enforced in access control.

* Shaded box denotes the preferred choice.

Table 2. Summary of Selection Process.

F. SUMMARY

This chapter described the process for the selection of wiki engines. TWiki was selected for implementation in the MYSEA testbed. The next chapter will discuss the approach and strategy used to implement wiki technology in the MYSEA environment.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. PROJECT DESCRIPTION

This chapter explains the concept of operation of the wiki engine in the MYSEA test bed environment, and describes the methodology for porting wiki technology to the MYSEA testbed.

A. CONCEPT OF OPERATION

In the MYSEA testbed environment, users need to be authenticated via the Trusted Path Extension (TPE) (also called the Trusted Computing Based Extension (TCBE)) before connection to any services in the MLS server is permitted. Following identification and authentication, users negotiate a session level from the range of security levels for which they are authorized. A user's maximum session level will be bounded by his or her clearance. The session level determines the level of access to data that are provided by the services in the MYSEA testbed, including the wiki, through the enforcement of MAC policy by the STOP 6.3 operating system.

Once login is completed, a user can access the wiki via any web browser through another wiki login. At this point, the wiki server enforces the wiki-specific DAC policy for the wiki session. The user can then read, edit or create wiki pages to which access is allowed, as constrained by the DAC policy set on the wiki server, as well as the MAC policy enforced by the STOP 6.3 operating system.

The following three example scenarios illustrate the possible collaboration between two users in a multilevel environment.

Example Scenario 1: Alice and Bob are logged in at the SIM_UNCLASSIFIED session level. Both of them can simultaneously edit wiki pages at that level. Alice decides that she wants to edit the "MeetingRoomBooking" page, and clicks on the "edit" link on the page to insert her schedule. Before she saves her changes, Bob decides that he wants to book the same meeting room, and clicks on the "edit" link for the same page. He is warned that Alice is currently editing the page, and may choose to proceed with his

editing. After both of them have saved their work, the changes to the page will be merged. If there is any conflict between the changes, the last person to save will be presented with the conflicting sections, and he or she will have to decide on how to resolve the conflict.

Example Scenario 2: Alice is logged in at the SIM_CONFIDENTIAL session level, while Bob is logged in with the SIM_UNCLASSIFIED session level. As Alice is the secretary of a coalition board meeting, she is uploading the meeting presentation materials, which are at the SIM_CONFIDENTIAL level. Alice is able to view, but not modify, information on the wiki labeled at the SIM_UNCLASSIFIED level. Bob is an invited attendee of the meeting, and he is browsing the meeting agenda page at the SIM_UNCLASSIFIED level. However, Bob will not be able to see the meeting presentation materials uploaded by Alice, which have a security level dominating his session level, and will not even know that the upload area exists.

Example Scenario 3: During the board meeting, Bob was asked to provide some documents to the committee. He logs in at the SIM_UNCLASSIFIED session level, and uploads the documents to the designated folder. Alice and the other board members who are logged in at the SIM_CONFIDENTIAL session level, will be able to read the documents uploaded by Bob.

B. METHODOLOGY

Following the selection of a wiki engine based on the selection process outlined in Chapter III, the implementation of the wiki engine on MYSEA testbed was completed in stages to aid in troubleshooting. The implementation began with the installation of the wiki engine on RedHat 8.0 Linux on an Intel-based machine, followed by its installation on the XTS-400 as a single-level process confined to a single-level domain, and finally the wiki was implemented on the XTS-400 server so that it could use the multilevel capabilities of the underlying operating system. During each stage, testing was performed to ensure that the wiki functionalities were preserved.

1. Linux

For the first stage, RedHat Linux 8.0 was installed as a virtual machine on a HP laptop running VMWare as the virtual machine manager. Virtual machine was used in order to reduce the amount of hardware resources needed and Redhat 8.0 was chosen because the STOP 6.3 operating system of the XTS-400 server provides an environment for the execution of RedHat 8.0 Linux-based application programs. The TWiki engine required the use of Perl scripting engine, and hence, Perl was installed on the virtual machine. For the web server, Apache 1.3.34 was installed as this was the version installed on the MYSEA MLS server. The Apache server was configured to be able to process the Perl scripts under the wiki directory. The purpose of this stage was to become familiar with the installation and configuration process of the required software components. Using the Apache and the wiki engine documentation, the software was set up without any issues. A few basic functional tests such as listing the wiki topic, displaying the topic's content, editing the topic, creating a new topic, deleting a topic, uploading attachments, and setting of access control permissions were performed to demonstrated the operation of wiki in a standard environment. After these tests were successfully performed, it was decided to proceed to the next stage of the implementation.

2. Single Level Mode on the XTS

The same versions of an Apache-like server (i.e., Apache modified to be MLS-aware) and the TWiki engine were installed to execute in single level mode on XTS server. As this stage, the Apache-like web sever was configured to run as a standalone server at a single security level. The security level was configured as *secrecy level 1* and *integrity level 3*, which is the security level configured for the network interface card. This is needed because in single level mode, all network traffic through the network interface card is labeled based on the security level configured for the card. The installation and configuration procedures were identical to those of Linux. The same basic functional tests were performed to ensure that the wiki worked as intended in the

new system. After the tests were completed successfully, the next stage was to adapt the wiki to run as a MLS-aware application on the MYSEA server.

3. Multilevel Mode on the XTS

In a multilevel environment, the Apache-like web server must be set up as an *inetd*-like service, bound to a specific port and be initiated by the MYSEA Secure Session Server (SSS) daemons. The SSS processes behave similarly to the standard *inetd* service in that they listen for new incoming connections and spawn new processes as appropriate to handle incoming requests. The difference is that the SSS processes spawn new process at the same security and integrity level as the level of the incoming request. To set up the multilevel environment, the MYSEA services were configured according to the MYSEA Version 2.0 instruction manuals. This was followed by TWiki configuration, the majority of which was identical to that of single level environment with some additional steps. The additional steps involved setting up two deflection directories for storage of temporary files and deleted files, where a deflection directory is a directory that exists independently at all security levels simultaneously (see Chapter II). The steps also involved creation of various wiki web directories at different security levels, and setting of the appropriate files and directory permissions. The detailed design and architecture is outlined in Chapter V.

C. SUMMARY

This chapter stated the goals of the project, described the concept of operation of the wiki, and explained the methodology of porting the TWiki engine to the MYSEA testbed. The following chapter will describe the architecture and design of TWiki in the context of MLS environment.

V. WIKI DESIGN AND ARCHITECTURE

This chapter outlines the architecture of TWiki for a standard setup and describes changes needed to implement it in the MYSEA MLS environment. The chapter also documents the issues encountered during the implementation phase and the proposed solutions to these problems.

A. STANDARD TWIKI ARCHITECTURE

In a standard TWiki setup, the contents of the wiki website are organized into *webs* and *topics*. A topic is a wiki page. Users can create new topics of their own, and edit existing topics according to DAC policy. A web is a collection of related topics. In terms of file and directory structures, a web is a sub-directory within the main *data* directory, and a topic is a file within the web sub-directory. For each topic file, there is a corresponding version control file that records the change history of the topic. Users can also upload and download files to and from the wiki website. These files are stored in the *pub* directory which is organized in a manner similar to the *data* directory in terms of its web sub-directories. Figure 6 shows the file and directory structure of a standard TWiki installation.

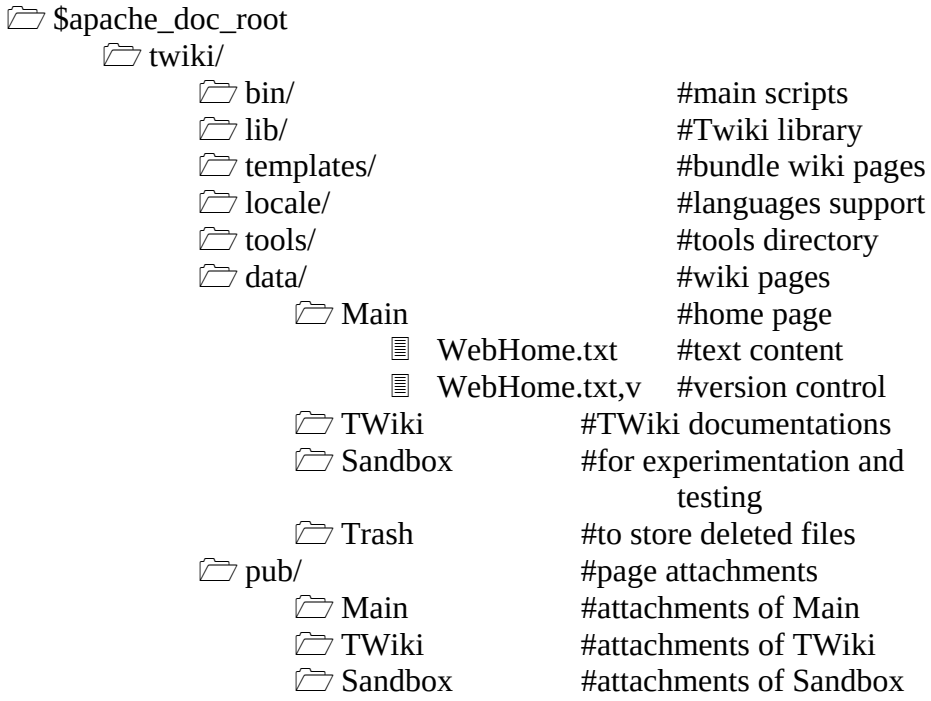

Figure 6. TWiki Directory Structure.

Within the various webs, sub-webs can be created in a hierarchical manner. The sub-webs are stored as sub-directories under the web directory. TWiki also requires a directory called “twiki” under the /tmp directory to store all temporary files, and a directory called “Trash” under the data directory to store deleted files. Figure 7 illustrates the directory structure of sub-web, where a sub-web called SubDirTest is created within the Main web.

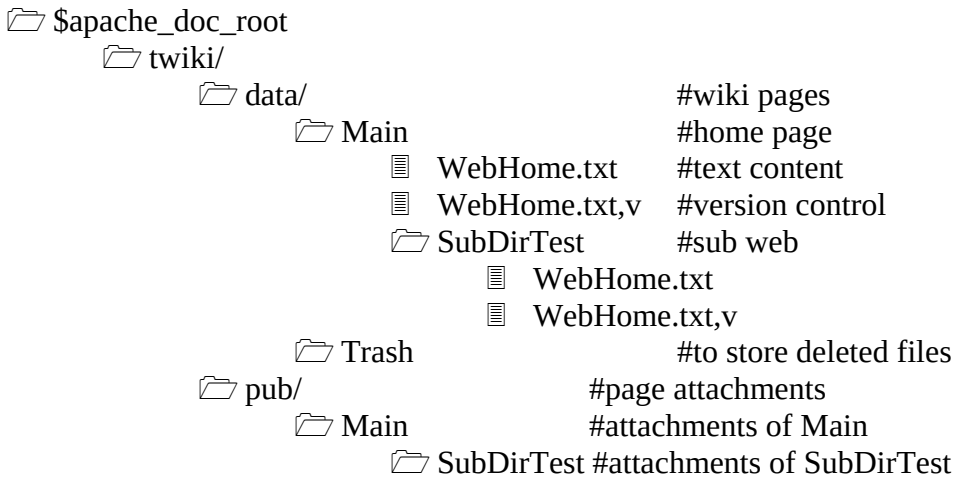

Figure 7. TWiki Sub-Web Directory Structure.

B. MLS WIKI ARCHITECTURE

When ported to the MLS environment, the twiki temporary file directory is created in the deflection directory of /tmp. This is needed because TWiki stores all its temporary files in this single /tmp/twiki directory. Creation of /tmp/twiki in a deflection directory ensures that users who login at different session levels will be able to write to the /tmp/twiki directory at their respective established levels. The Trash directory is also created as a deflection directory for the same reason, because TWiki stores all deleted files in a single Trash directory, and therefore a deflection directory would permit file deletion at all session levels. As an example, if a user login at the SIM_UNCLASSFIED level, the user would be able to see the SIM_UNCLASSIFIED directory and the Trash directory at SIM_UNCLASSIFIED. If another user login at the SIM_SECRET level, that user would be able to see the SIM_UNCLASSIFIED directory, the SIM_SECRET directory, and also the Trash directory at SIM_SECRET. Note that since the Trash directory is created as a deflection directory, the two users would only see the files deleted at their established session level.

As described in Chapter III, TWiki is able to support the organization of wiki content by classification or by topic. For organization by classification, webs corresponding to the various security classification levels are created under the *data* and

pub directories at the respective MAC levels. Sub-webs are then created within each of these webs to categorize different wiki topics. For organization by topic, webs corresponding to each topic of interest are created under the *data* and *pub* directories at the lowest MAC level (i.e., SIM_UNCLASSIFIED). Sub-webs are then created within each of these content webs at the corresponding MAC levels. In this approach, all the possible login session levels must be anticipated and the corresponding web directories created in advance. Sub-webs within these web directories can be created when required by the administrator.

Figures 8 and 9 illustrate the directory structure for organization by classification and organization by topic respectively.

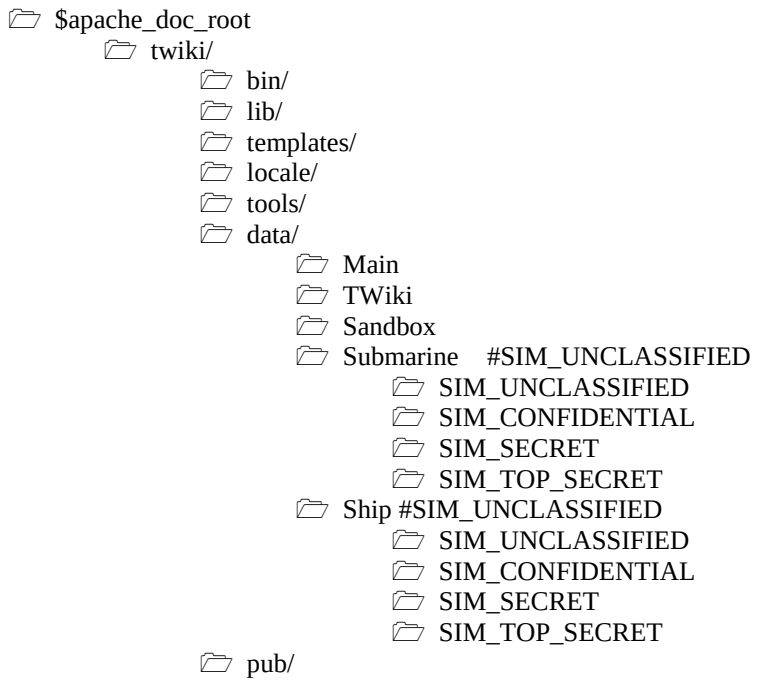

Figure 8. Organization by Topic.

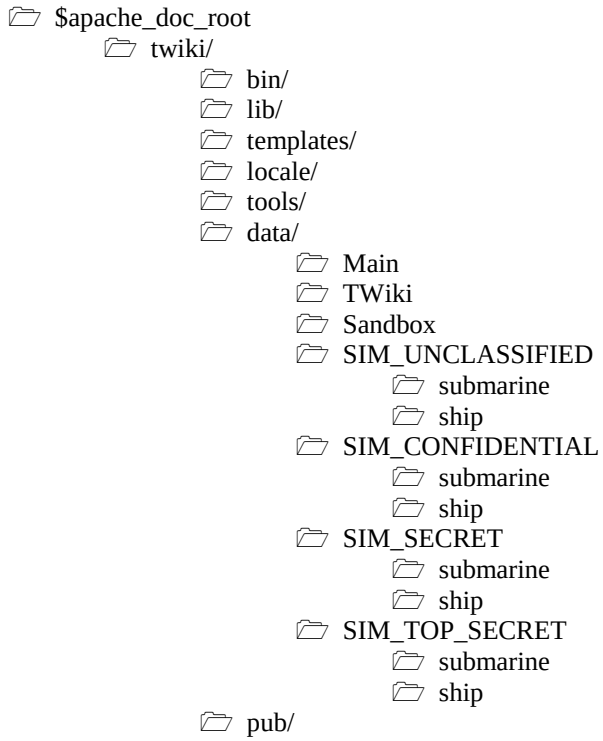

Figure 9. Organization by Classification.

In MLS wiki, users are able to read content with security label equal to or below the currently established session level, but only write content with security label equal to the currently established session level. If a user inserts a link to a wiki page with a security level higher than the session level, the creation of the link will succeed. This is because link creation only requires read and write permissions to the working wiki page. TWiki will show a question mark "?" beside the newly created link to indicate that the target page does not exist. This is the standard TWiki behavior for creating links to any non-existent wiki pages, regardless of whether the target page is at the same security level or different security level. If the user attempts to access the newly created link, the TWiki engine will inform the user that the higher security level web does not exist and provide an option to create the web, regardless of whether the link is to a new page or an existing page in the higher security level web. If the user attempts to create the web, the operation will be denied, as writeup operation is not permitted on the server.

C. WIKI ISSUES IN MYSEA

In the multilevel MYSEA environment, the Apache-like web server runs as part of the MYSEA services, which listens to incoming requests and spawns new processes running at the same security and integrity levels as the negotiated session level. The spawned web server processes also take on the session's user ID and a specific group ID, causing different wiki topics created by different users to be owned by its creator. In the standard TWiki configuration, this will not work as the TWiki engine expects the Apache server to run as a single user, which is commonly named as *apache*, such that all wiki files are readable and writeable only to the *apache* user. Hence, only the Apache process can read and write to files and directories created by the TWiki engine. The TWiki engine then enforces its own access control mechanism to allow TWiki users to set permissions on the files they have created. In the MYSEA environment, since the wiki files created by the web server processes are owned by their creators, all files, including the newly created files, must have read/write permissions set for the group called *other* which is used by the web server. Users must belong to the same group so that they can access the files at the OS level. TWiki then enforces the same application-level access control mechanism as in the case of its standard implementation. To implement the required changes, an attempt was made to change the default permission of new text files to 664 (i.e., rw-rw-r--) by setting the *umask* to 002 in the /etc/profile script. This works for the interactive console logon session, but does not work for the Apache daemon process as it does not inherit the *umask* setting from the /etc/profile script. It was therefore decided that the TWiki code had to be modified to include a *umask* command to change the default permission on new files. Upon modification of the code, the default permission on each new wiki file is set to 664.

However, this introduced a vulnerability, as users could login through an interactive shell session and by-pass the TWiki access control to gain direct access to the wiki files. For example, a user could establish an interactive session through a program

such as WebShell to the MYSEA server, and view or modify the content of wiki files using an OS command such as *cat*. The following sections describe possible solutions to overcome this limitation.

D. SOLUTIONS TO THE TWIKI DAC PROBLEM

To overcome the limitation introduced by the modification of default permissions on new TWiki files, two different approaches were investigated. The first approach involves restricting the users' ability to access files through interactive shell sessions, whereas the second approach involves the use of the XTS access control list functionality to restrict file access.

1. Restricting Interactive Shell Sessions

To remove the vulnerability of DAC policy enforcement by-pass and resulting direct file access, the user's ability to launch interactive shell session must be restricted. This can done by ensuring that services such as SSH and telnet are not enabled. In the MYSEA testbed, SSH and *telnet* services are not supported, but users are allowed to launch interactive shell sessions via the WebShell CGI program. Though the WebShell program does not allow the execution of interactive programs such as the *vi* editor, users can still modify the contents of a file by other mechanisms such as output redirection. As an example, the command "echo text > myFile" would replace the contents of myFile with "text".

The WebDAV implementation in MYSEA allows users to navigate to their home directory via a symbolic link to the /home directory under the WebDAV document root. This also allows users to navigate to the Apache document root, as it is located under /home/http/htdocs which is under the /home directory. The users can therefore view and edit files under the TWiki data directory using WebDAV.

As the use of WebShell and WebDAV are part of the overall concept of operations of MYSEA, removing these services on the MYSEA server is not a viable

solution. However, a separate server can be implemented to only provide wiki services. Currently such federation of servers would mean users are required to sign-in more than once, but studies investigating single-single-on for the MYSEA testbed have been done and a framework for single-sign-on for MYSEA has been proposed [15]. In the separate wiki server, WebShell and WebDAV will be disabled, thus removing the direct file access capability. For WebShell, this is done by removing the CGI program from the cgi-bin directory under the Apache directory. The removal of WebDAV requires recompilation of the Apache source code. Another alternative to overcome the WebDAV vulnerability in the separate wiki server is to restructure the user home directory such that it is located under the WebDAV document root so that the symbolic link to /home can be removed. However, this would mean that there is no standardization in the directory structure of the servers in the MYSEA testbed as the WebDAV repository on the wiki server will be mounted on a different directory root compared to the WebDAV repository on other servers, and it is therefore not recommended.

2. XTS Access Control List

The STOP 6.3 operating system supports the use of Access Control Lists (ACL). Up to seven entries can be added to the ACL for each file or directory, and each of these entries can be a combination of user names and group names. The insertion of an entry to the ACL is done through the `fsm` command [16].

A subset of the TWiki access modes can be mirrored using this ACL feature. TWiki supports three access modes: *view*, *change* and *rename* for viewing, changing and renaming topics or webs, respectively. The *view* access mode corresponds to the read access mode of the STOP OS, whereas the *change* and *rename* access modes correspond to the write access mode. Therefore, certain granularity will be lost if the XTS ACL mechanism is adopted to implement the default TWiki DAC mechanism, since the change and rename modes are mapped to the same write permission. Furthermore, the revision control file, which is used to generate the history log, will need to have the same entries added in the ACL as the topic file itself. This would mean that any authorized users of the wiki topic can also modify the history log. The TWiki password file where

the hashes of the user passwords are stored will have this vulnerability, although this could be mitigated through procedural controls by disallowing password change by the user. Specifically, this can be achieved by changing the file ownership of the password file to be owned by the administrator.

The ACLs can be implemented at the directory level or file level. At the directory level, authorized users are added to the ACL of the web directory with read and write permission through the *fsm* command during the web creation process which can only be done by the administrator. All files within the directory have owner and group read/write permissions enabled. The effect of this is that the user will either have read and write permission to all files or to none. It is not possible to have just read permission, as that will only restrict the user in terms of disallowing the creation of new file within the directory, but the user will still be able to modify existing files.

At the file level, authorized users are added to the ACLs of each topic file and to its corresponding version control file. As users have ability to create their own topics, the number of topic files and the associated version control files can be large. It is therefore not practical to burden the administrator with the task of adding authorized users to the ACLs. Development of new code for insertion and deletion of ACL entries as well as modification of the standard TWiki scripts to interface with the newly developed codes would therefore be necessary.

3. Proposed Solution

The implementation of ACLs at the granularity of the file level for TWiki is not recommended for the MYSEA testbed because of the requirement for customization of the wiki code. Customization of the wiki code means that the solution is less portable in terms of wiki engine maintenance or replacement. The implementation of ACLs at directory level for TWiki is also not recommended because the files within the directory would need to be group readable and writeable which causes the password file and history log files to be vulnerable to misuse. Eliminating interactive shell sessions on a separate server dedicated for wiki services is therefore adopted for this study. This

solution preserves all wiki functionality and at the same time allows for wiki engine upgrades or changes with minimal future modification.

E. IMPLEMENTATION

To implement the proposed solution in the MYSEA environment, a second XTS-400 server was set up to serve as the standard MYSEA MLS server to test the proposed federated design. The test configuration is shown in Chapter VI.

The directory structure of the wiki server was organized by classification. Several directories corresponding to the security levels defined on the MYSEA server were created. Figure 10 shows the directory structure within the data directory with the security label of the various sub-directories denoted using comments after the # symbol. . The pub directory was configured with a similar directory structure.

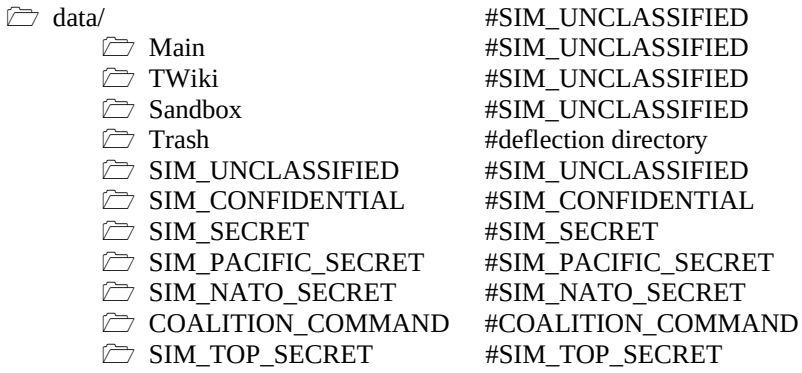

Figure 10. Directory Structure.

The data and pub directories, and all the sub-directories within them, were created with file permissions of 775, i.e., owner and group have full permissions, while others only have read and execute permissions. This allows existing files to be read and new files to be created by members belonging to the group. All files under data and pub have file permissions of 664, i.e., owner and group have read and write permissions while others only have read permission.

In order for new webs and topics created by the wiki engine to have the same permission settings described above, the source code of the wiki engine was modified as described in Section C of this chapter. In particular, a "umask=002" command was inserted in two Perl modules of the TWiki engine, namely manage.pm and save.pm, which contain code to create new webs and new topics, respectively.

In terms of server configuration, the WebDAV module was removed via recompilation of Apache source code. The WebShell GGI script was also removed from the cgi-bin directory so that users can no longer execute any interactive shell session on the wiki server. The server configuration procedures are described in Appendix A.

F. SUMMARY

This chapter described the design of a standard TWiki installation, and the design and architecture of the TWiki integration into the MYSEA environment. The issues encountered were discussed, and various solutions were presented. The proposed solution is to implement the wiki engine on a separate server with the interactive shell session disabled. The next chapter will describe the testing procedures and the results of the tests.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. TESTING AND RESULTS

This chapter describes the test plan and the testing results for the use of TWiki in the MYSEA testbed. There are four parts to the testing: functional test, DAC security test, MAC security test and integration test. Functional testing is required to confirm the expected functionalities of the wiki engine and provides a baseline measurement on the ability of the system to perform its task under a specific environment. The functional tests performed fall into three configuration categories: Linux, single level XTS and multilevel XTS. DAC security tests were performed for all configurations to verify that the DAC policies are being enforced correctly. MAC security tests were also performed for the single level XTS and multilevel XTS configurations to verify that the MAC policies are being enforced correctly. Integration testing is required to verify that the federated wiki server is able to function correctly in the MYSEA environment, and that users are able to use MYSEA services available on both servers.

The test topology consisted of a simple standalone network with two XTS-400 servers, a Windows XP client desktop and a Windows XP client laptop, all connected via a network hub. Both client machines are capable of running virtual machines through the VMWare Server installed on the clients. Testing was done through the Internet Explorer browser on the Windows XP client and the FireFox browser on the Linux virtual machine running within the VMWare Server on the Windows XP client. Figure 11 shows the network topology of the test setup.

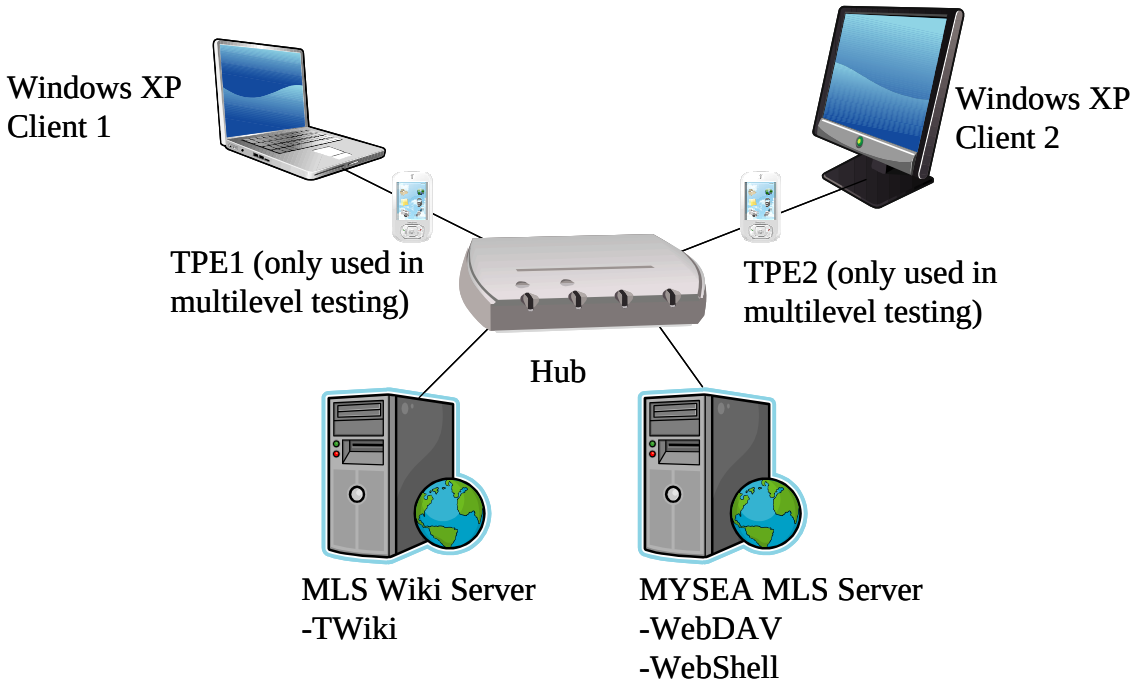

Figure 11. Network Topology of Test Setup.

A. FUNCTIONAL TEST PLAN

Functional tests were performed on the three test configurations (Linux, single level XTS, and multilevel XTS). The tests were done to ensure that the wiki engine work as intended in the specific environment. The functional tests can be broadly categorized into three types: read, write and administration. The read tests involve reading wiki pages through the TWiki engine via the list, display and download capabilities. The write tests involve writing wiki pages through the TWiki engine via the edit, create, delete and upload functions. Administration type tests involve performing administrative tasks. These tests include configuring the TWiki engine, setting permissions of the wiki contents, and registering new user account. Two TWiki test user accounts, TestUser1 and TestUser2 were used for concurrent testing in test cases A1 to A9. The objective is to ensure that two users logged in from two client machines can view and modify different files at the same time. Table 3 lists the various functional tests that were performed.

Test #	Test Type	Test Objective	DAC Setting	MAC Setting	Expected Result for: TestUser1	TestUser2
A1	List	Ensure that user can list wiki pages	allow	allow	pass	pass
A2	Display	Ensure that user can display wiki pages	allow	allow	pass	pass
A3	Download	Ensure that user can download wiki pages	allow	allow	pass	pass
A4	Edit	Ensure that user can edit wiki pages	allow	allow	pass	pass
A5	Create	Ensure that user can create wiki pages	allow	allow	pass	pass
A6	Delete	Ensure that user can delete wiki pages	allow	allow	pass	pass
A7	Upload	Ensure that user can upload wiki pages	allow	allow	pass	pass
A8	Simultaneous Edit	Ensure that multiple users can safely edit same wiki page concurrently	allow	allow	pass	pass
A9	Setting Permission	Ensure that user can change access rights of wiki pages	allow	allow	pass	pass
A10	Configure	Ensure that administrator can configure Twiki	allow	allow	pass	pass
A11	Register User	Ensure that guest can register for login account	allow	allow	pass	pass

Table 3. List of Functional Tests.

B. LINUX TESTING

The functional tests performed on Linux served as a baseline for the functionality for a TWiki server. In other words, if a functionality does not work in the Linux environment, it is not expected to work in the XTS environment.

1. TWiki DAC Test Plan

TWiki DAC tests were performed to ensure that the TWiki DAC policies were being enforced correctly. The TWiki DAC test plan includes test cases involving the three permission modes enforced by TWiki: *view*, *change* and *rename* modes. For each mode there are four settings: DenyTopic, AllowTopic, DenyWeb and AllowWeb. Table 4 shows the permitted values and the processing rules for each of the settings.

Setting	Permitted values	Processing Rules
DenyTopic	not set (i.e. deleted or commented out)	not evaluated
	empty	no one is denied
	comma-delimited list of Users and Groups	people in the list will be denied
AllowTopic	not set (i.e. deleted or commented out)	not evaluated
	empty	equivalent to not setting
	comma-delimited list of Users and Groups	people in the list will be allowed, people NOT in the list will be denied
DenyWeb	not set (i.e. deleted or commented out)	not evaluated
	empty	equivalent to not setting
	comma-delimited list of Users and Groups	people in the list will be denied
AllowWeb	not set (i.e. deleted or commented out)	not evaluated
	empty	equivalent to not setting
	comma-delimited list of Users and Groups	people in the list will be allowed, people NOT in the list will be denied

Table 4. Permitted Values and Processing Rules of TWiki DAC Settings.

From Table 4, it can be seen that for DenyTopic setting, there are three possible permitted values (not set, empty and list of users/group). For each of the other three settings (AllowTopic, DenyWeb and AllowWeb) there are two possible permitted values, since an empty value is equivalent to not setting any value. Therefore, there are 24 different possible setting combinations involving the *view*, *change* and *rename* permission modes.

In evaluating the DAC settings, note that TWiki DAC enforcement mechanism adheres to the rules showed in Table 5 in the listed order (i.e., evaluation starts from the top of the list; if a rule is met, then it is applied immediately and no other rules are considered). The three modes: *view*, *change* and *rename* may be granted or denied separately [17].

S/N	TWiki DAC Rules	Action
1	If the user is an administrator	access is **PERMITTED**.
2	If DENYTOPIC is set to a list of user names	people in the list will be **DENIED**.
3	If DENYTOPIC is set to *empty* (i.e. Set DENYTOPIC =)	access is **PERMITTED** *i.e* no-one is denied access to this topic..
4	If ALLOWTOPIC is set	people in the list are **PERMITTED,** everyone else is **DENIED**
5	If DENYWEB is set to a list of user names	people in the list are **DENIED** access
6	If ALLOWWEB is set to a list of wikinames	people in the list will be **PERMITTED**, everyone else will be **DENIED**
7	If none of the above rules match	access is **PERMITTED**.

Table 5. Order of TWiki DAC Rule Evaluation (After [17]).

In formulating the DAC test plan, all the 24 possible combinations of setting the TWiki DAC were considered. Two additional scenarios were also considered: one for explicitly testing administrator permissions, i.e., Rule 1 of Table 5, and the other to ensure TWiki DAC does not override OS DAC. Table 6 shows an example of the 26 different scenarios. Six out of the 26 scenarios were shortlisted as candidates to verify that both the processing rules and the order of rule evaluation are being enforced as stated in Table 4 and Table 5. The six scenarios were selected such that when a particular mode is set (e.g., DenyView), the next mode with an opposite effect (e.g., AllowView) at a lower order of evaluation would be set. This is to ensure that TWiki stops evaluating other rules once a rule is matched, and any other lower priority rules have no effect. For example, in Test Scenario 4 (S/N. 10) AllowTopic and DenyWeb are both set to TestUser1. From Table 5, TestUser1 will match Rule 4 (AllowTopic), and will be granted access. Although DenyWeb is also set to TestUser1, it should have no effect as it will not be evaluated by TWiki. This approach verifies that the order of rule evaluation is correctly enforced, and the six scenarios ensure that Rules 2 to 7 of Table 5 are verified. Besides the six scenarios, the two scenarios for testing of administrator permissions, i.e., to verify Rule 1 (S/N. 25 of Table 6), and testing that TWiki DAC does not override OS DAC (S/N. 26 of Table 6) were also considered. Out of these eight candidate scenarios,

five exception cases where users are expected to be denied access were chosen as final test scenarios. The other three candidates were not chosen as test scenarios because all their results are expected to pass.

For the five selected test scenarios, test cases for the *view*, *change* and *rename* modes were formulated for the Linux, single level, and mulitlevel configurations. The view, change and rename test cases were distributed among the five test scenarios. The results from the Linux test cases were used for establishing a baseline for the expected results. In each of the test cases, three test instances, each involving a different user accounts were performed. The user accounts were: TestUser1, TestUser2 and AdminUser. TestUser2 was mainly used to verify the processing rules of AllowTopic and AllowWeb, where a user listed in the rules (i.e., TestUser1) is allowed access and all others (i.e., TestUser2) are denied. These test cases are designed to verify that the TWiki DAC policies are enforced in accordance to TWiki documentation. Table 7 shows the various test scenarios and test cases selected for testing, and the total number of test instances performed. Windows Internet Explorer client was used for the test cases of the first three scenarios, while Linux FireFox client was used for test cases of the last two scenarios. A total of 57 test instances comprised the TWiki DAC testing.

S/N	TWiki DAC: VIEW/CHANGE/RENAME Permission Mode Setting				Expected Results for Subject *:			Remarks
	AllowWeb	DenyWeb	AllowTopic	DenyTopic	TestUser1	TestUser2	AdminUser	
1	-	-	-	-	Pass (7)	Pass (7)	Pass (1)	Not selected as all are expected to pass.
2	-	-	-	TestUser1	Fail (2)	Pass (7)	Pass (1)	
3	-	-	-	empty	Pass (3)	Pass (3)	Pass (1)	
4	-	-	TestUser1	-	Pass (4)	Fail (4)	Pass (1)	
5	-	-	TestUser1	TestUser1	Fail (2)	Fail (4)	Pass (1)	Selected as Test Scenario 1 to test the exception.
6	-	-	TestUser1	empty	Pass (3)	Pass (3)	Pass (1)	
7	-	TestUser1	-	-	Fail (5)	Pass (7)	Pass (1)	
8	-	TestUser1	-	TestUser1	Fail (2)	Pass (7)	Pass (1)	
9	-	TestUser1	-	empty	Pass (3)	Pass (3)	Pass (1)	Not selected as all are expected to pass.
10	-	TestUser1	TestUser1	-	Pass (4)	Fail (4)	Pass (1)	Selected as Test Scenario 2 to test the exception.
11	-	TestUser1	TestUser1	TestUser1	Fail (2)	Fail (4)	Pass (1)	
12	-	TestUser1	TestUser1	empty	Pass (3)	Pass (3)	Pass (1)	
13	TestUser1	-	-	-	Pass (6)	Fail (6)	Pass (1)	Selected as Test Scenario 3 to test the exception.
14	TestUser1	-	-	TestUser1	Fail (2)	Fail (6)	Pass (1)	
15	TestUser1	-	-	empty	Pass (3)	Pass (3)	Pass (1)	
16	TestUser1	-	TestUser1	-	Pass (4)	Fail (4)	Pass (1)	
17	TestUser1	-	TestUser1	TestUser1	Fail (2)	Fail (4)	Pass (1)	
18	TestUser1	-	TestUser1	empty	Pass (3)	Pass (3)	Pass (1)	
19	TestUser1	TestUser1	-	-	Fail (5)	Fail (6)	Pass (1)	Selected as Test Scenario 4 to test the exception.
20	TestUser1	TestUser1	-	TestUser1	Fail (2)	Fail (6)	Pass (1)	
21	TestUser1	TestUser1	-	empty	Pass (3)	Pass (3)	Pass (1)	
22	TestUser1	TestUser1	TestUser1	-	Pass (4)	Fail (4)	Pass (1)	
23	TestUser1	TestUser1	TestUser1	TestUser1	Fail (2)	Fail (4)	Pass (1)	
24	TestUser1	TestUser1	TestUser1	empty	Pass (3)	Pass (3)	Pass (1)	
25		AdminUser		AdminUser	Pass (7)	Pass (7)	Pass (1)	Not selected as all are expected to pass.
26			Testuser1& 2		Fail (NA)	Fail (NA)	Fail (NA)	Selected as Test Scenario 5 to test the exception.

* number in bracket denotes active rule for the subject.

" - " denote not set.

Table 6. Possible Combination of TWiki DAC Setting.

a. Test Scenarios	b. Configurations	c. Permission Modes to be Tested	d. Number of Test Cases in c.	e. Number of Subj per Test Case	f. Number of Test Instance (i.e. done by one subj) per Scenario per Configuation (d.*c.)	g. Type of Client Tested
1	Linux	V	1	3	3	
	Single Level	V(E)	1	3	3	
	Multilevel	V(E)	1	3	3	
2	Linux	V	1	3	3	
	Single Level	V(D)	1	3	3	IE
	Multilevel	V(D)	1	3	3	
3	Linux	C	1	3	3	
	Single Level	C	1	3	3	
	Multilevel	C	1	3	3	
4	Linux	R	1	3	3	
	Single Level	R	1	3	3	
	Multilevel	R	1	3	3	FF
5	Linux	V,C,R	3	3	9	
	Single Level	V(E),C	2	3	6	
	Multilevel	V(D),R	2	3	6	
Total Number of Test Instance to be Performed					**57**	

V - View
C - Change
R - Rename
V(E) - View at Read Equal (only for XTS 400)
V(D) - View at Read Down (only for XTS 400)
FF - FireFox
IE - Internet Explorer

Table 7. Test Scenarios and Test Cases Performed.

Based on the test cases identified, TWiki DAC test plans were formulated for the view, change and rename permission mode. Except for test cases BL4, BL5 and BL6, the default OS DAC for the TWiki test files used in other test cases were set with standard Unix permission bits [22] of 664, i.e., read/write for owner and group, and read only for others. For test case BL4, permission bits of the file were set to 000, i.e. nobody has read or write permission. For test case BL5, permission bits of the file were set to 444, i.e., everyone has read-ony permission. For test case BL6, permission bits of the directory

containing the file were set to 555, i.e. nobody can write to the directory and hence the files within the directory cannot be renamed. Table 8 shows the various TWiki DAC tests that were performed.

Test #	Test Type	TWiki DAC: VIEW Permission Mode Setting				Expected Results for Subject:		
		AllowWeb	DenyWeb	AllowTopic	DenyTopic	TestUser1	TestUser2	AdminUser
BL1	TWiki DenyTopicView	-	-	TestUser1	TestUser1	Fail	Fail	Pass
	Tests BL2 ensures that a user is not able to read wiki page that has access denied through DenyTopicView.							
BL2	TWiki AllowTopicView	-	TestUser1	TestUser1	-	Pass	Fail	Pass
	Test BL4 ensures that a user is able to read wiki page that has access granted through AllowTopicView.							
BL3	TWiki AllowWebChange	TestUser1	-	-	-	Pass	Fail	Pass
	Test BL6 ensures that a user is able to read wiki page with access granted through AllowWebView.							
BL4	TWiki DenyWebRename	TestUser1	TestUser1	-	-	Fail	Fail	Pass
	Test BL5 ensures that a user is not able to read wiki page that has access denied through DenyWebView.							
BL5	TWiki View DAC Not overriding OS DAC	-	-	TestUser1, TestUser2	-	Fail	Fail	Fail
	Test BL5 ensures that TWiki View DAC does not overwrite OS DAC (file permission bit set to 000)							
BL6	TWiki Change DAC Not overriding OS DAC	-	-	TestUser1, TestUser2	-	Fail	Fail	Fail
	Test BL6 ensures that TWiki Change DAC does not overwrite OS DAC (file permission bit set to 444)							
BL7	TWiki Rename DAC Not overriding OS DAC	-	-	TestUser1, TestUser2	-	Fail	Fail	Fail
	Test BL7 ensures that TWiki Rename DAC does not overwrite OS DAC (directory permission bit set to 555)							

Table 8. Linux TWiki DAC Test.

2. Test Results

All tests performed successfully. Table 9 shows the results of the Linux functional tests.

Test #	Test Type	Internet Explorer		FireFox	
		TestUser1	TestUser2	TestUser1	TestUser2
A1	List	pass	pass	pass	pass
A2	Display	pass	pass	pass	pass
A3	Download	pass	pass	pass	pass
A4	Edit	pass	pass	pass	pass
A5	Create	pass	pass	pass	pass
A6	Delete	pass	pass	pass	pass
A7	Upload	pass	pass	pass	pass
A8	Simultaneous Edit	pass	pass	pass	pass
A9	Setting Permission	pass	pass	pass	pass
A10	Configure	pass	pass	pass	pass
A11	Register User	pass	pass	pass	pass

Table 9. Linux Server Functional Test Results.

Table 10 shows the results of the Linux TWiki DAC testing. The results show that all the wiki operations on the clients behaved as expected when constrained by the TWiki DAC policy.

Test #	Test Type	Internet Explorer			FireFox		
		TestUser1	TestUser2	AdminUser	TestUser1	TestUser2	AdminUser
BL1	TWiki DenyTopicView	Fail	Fail	Pass	Fail	Fail	Pass
BL2	TWiki AllowTopicView	Pass	Fail	Pass	Pass	Fail	Pass
BL3	TWiki AllowWebChange	Pass	Fail	Pass	Pass	Fail	Pass
BL4	TWiki DenyWebRename	Fail	Fail	Pass	Fail	Fail	Pass
BL5	TWki View DAC not overriding OS DAC	Fail	Fail	Fail	Fail	Fail	Fail
BL6	TWki Change DAC not overriding OS DAC	Fail	Fail	Fail	Fail	Fail	Fail
BL7	TWki Rename DAC not overriding OS DAC	Fail	Fail	Fail	Fail	Fail	Fail

Table 10. Linux Server DAC Test Results.

C. SINGLE LEVEL XTS TESTING

The tests outlined in this section were performed on the single level XTS server. The objective was to ensure that no functionality was lost during the transition from Linux server to the single level XTS server, and that the DAC and MAC security constraints were met.

1. TWiki DAC Test Plan – Single Level Configuration

Another set of TWiki DAC tests were performed to ensure that the TWiki DAC policies were being enforced correctly in the single level XTS configuration. The security labels of the test subjects and objects were set such that all test operations were allowed with respect to MAC policy. Table 11 shows the various TWiki DAC tests that were performed.

Test #	Test Type	TWiki DAC: VIEW Permission Mode Setting				Subject Level	Object Level	Operation	Expected Results for Subject:		
		AllowWeb	DenyWeb	AllowTopic	DenyTopic				TestUser1	TestUser2	AdminUser
BS1	TWiki DenyTopicView	-	-	TestUser1	TestUser1	sl1:il3	sl1:il3	Read equal	Fail	Fail	Pass
	Tests BS1 ensures that a user is not able to read wiki page that has access denied through DenyTopicView.										
BS2	TWiki AllowTopicView	-	TestUser1	TestUser1	-	sl1:il3	sl0:il3	Read down	Pass	Fail	Pass
	Test BS2 ensures that a user is able to read wiki page that has access granted through AllowTopicView.										
BS3	TWiki AllowWebChange	TestUser1	-	-	-	sl1:il3	sl1:il3	Write equal	Pass	Fail	Pass
	Test BS3 ensures that a user is able to read wiki page with access granted through AllowWebView.										
BS4	TWiki DenyWebRename	TestUser1	TestUser1	-	-	sl1:il3	sl1:il3	Write equal	Fail	Fail	Pass
	Test BS4 ensures that a user is not able to read wiki page that has access denied through DenyWebView.										
BS5	TWiki View DAC Not overriding OS DAC	-	-	TestUser1, TestUser2	-	sl1:il3	sl1:il3	Read equal	Fail	Fail	Fail
	Test BS5 ensures that TWiki View DAC does not overwrite OS DAC (file permission bit set to 000)										
BS6	TWiki Change DAC Not overriding OS DAC	-	-	TestUser1, TestUser2	-	sl1:il3	sl1:il3	Write equal	Fail	Fail	Fail
	Test BS6 ensures that TWiki Change DAC does not overwrite OS DAC (permission bit set to 444)										

Table 11. Single Level XTS TWiki DAC Test.

2. MAC Test Plan – Single Level Configuration

MAC tests were performed to ensure that the MAC policies were enforced correctly. The MAC test cases involve reading and writing at different security levels. Since every user running on the single level XTS server is bound to a single security level, i.e., sl1:il3, only variance in the object security level can be tested. Table 12 shows the MAC tests that were performed.

Test #	Test Type	TWiki DAC: VIEW & CHANGE Permission Mode Setting				Subject Level	Object Level	Operation	Expected Result for Subject: TestUser1
		Allow Web	Deny Web	Allow Topic	DenyTopic				
CS1	Secrecy Read Up	-	-	TestUser1	-	sl1:il3	sl2:il3	Read	Fail
	Test CS1 ensures that a user is not able to read wiki page at a secrecy level higher than the level of the currently established session.								
CS2	Secrecy Read Down	-	-	TestUser1	-	sl1:il3	sl0:il3	Read	Pass
	Test CS2 ensures that a user is able to read wiki page at a secrecy level lower than the level of the currently established session.								
CS3	Secrecy Read Equal	-	-	TestUser1	-	sl1:il3	sl1:il3	Read	Pass
	Test CS3 ensures that a user is able to read wiki page at a secrecy level equal to the level of the currently established session.								
CS4	Secrecy Write Up	-	-	TestUser1	-	sl1:il3	sl2:il3	Write	Fail
	Test CS4 ensures that a user is not able to write to wiki page at a secrecy level higher than the level of the currently established session.								
CS5	Secrecy Write Down	-	-	TestUser1	-	sl1:il3	sl0:il3	Write	Fail
	Test CS5 ensures that a user is not able to writeto wiki page at a secrecy level lower than the level of the currently established session.								
CS6	Secrecy Write Equal	-	-	TestUser1	-	sl1:il3	sl1:il3	Write	Pass
	Test CS6 ensures that a user is able to write to wiki page at a secrecy level equal to the level of the currently established session.								

Table 12. Single Level MAC Test.

3. Test Results – Single Level Configuration

Table 13 shows that the results of the functional test on single level XTS were the same as those of Linux server. This shows that no functionality was lost during the transition to single level XTS.

Test #	Test Type	Internet Explorer		FireFox	
		TestUser1	TestUser2	TestUser1	TestUser2
A1	List	pass	pass	pass	pass
A2	Display	pass	pass	pass	pass
A3	Download	pass	pass	pass	pass
A4	Edit	pass	pass	pass	pass
A5	Create	pass	pass	pass	pass
A6	Delete	pass	pass	pass	pass
A7	Upload	pass	pass	pass	pass
A8	Simultaneous Edit	pass	pass	pass	pass
A9	Setting Permission	pass	pass	pass	pass
A10	Configure	pass	pass	pass	pass
A11	Register User	pass	pass	pass	pass

Table 13. Single Level XTS Functional Test Results.

Table 14 shows the results of the single level XTS TWiki DAC testing. The results show that all the wiki operations on the clients behaved as expected when constrained by the TWiki DAC policy.

Test #	Test Type	Internet Explorer			FireFox		
		TestUser1	TestUser2	AdminUser	TestUser1	TestUser2	AdminUser
BS1	TWiki DenyTopicView	Fail	Fail	Pass	Fail	Fail	Pass
BS2	TWiki AllowTopicView	Pass	Fail	Pass	Pass	Fail	Pass
BS3	TWiki AllowWebChange	Pass	Fail	Pass	Pass	Fail	Pass
BS4	TWiki DenyWebRename	Fail	Fail	Pass	Fail	Fail	Pass
BS5	TWki View DAC not overriding OS DAC	Fail	Fail	Fail	Fail	Fail	Fail
BS6	TWki Change DAC not overriding OS DAC	Fail	Fail	Fail	Fail	Fail	Fail

Table 14. Single Level XTS TWiki DAC Test Results.

Table 15 shows the results of the single level XTS MAC testing. The results show that all the clients behaved as expected when constrained by the MAC policy.

Test #	Test Type	Internet Explorer	FireFox
CS1	Secrecy Read Up	Fail	Fail
CS2	Secrecy Read Down	Pass	Pass
CS3	Secrecy Read Equal	Pass	Pass
CS4	Secrecy Write Up	Fail	Fail
CS5	Secrecy Write Down	Fail	Fail
CS6	Secrecy Write Equal	Pass	Pass

Table 15. Single Level XTS MAC Results.

D. MULTILEVEL XTS TESTING

The tests outlined in this section were performed on the XTS server operating as a multilevel MYSEA server. The objective was to ensure that no functionality was lost during the transition from the single level XTS configuration to the multilevel MYSEA configuration, and that the TWiki DAC and MAC security constraints were met.

1. TWiki DAC Test Plan – MLS Configuration

Another set of TWiki DAC tests was performed to ensure that the TWiki DAC policies were being enforced correctly in the multilevel MYSEA configuration. Table 16 shows the various TWiki DAC tests that were performed.

Test #	Test Type	TWiki DAC: VIEW Permission Mode Setting				Session Level	Object Level	Operation	Expected Results for Subject:		
		AllowWeb	DenyWeb	AllowTopic	DenyTopic				TestUser1	TestUser2	AdminUser
BM1	TWiki DenyTopicView	-	-	TestUser1	TestUser1	SIM_UNCLASSIFIED	SIM_UNCLASSIFIED	Read equal	Fail	Fail	Pass
	Tests BM1 ensures that a user is not able to read wiki page that has access denied through DenyTopicView.										
BM2	TWiki AllowTopicView	-	TestUser1	TestUser1	-	SIM_SECRET	SIM_UNCLASSIFIED	Read down	Pass	Fail	Pass
	Test BM2 ensures that a user is able to read wiki page that has access granted through AllowTopicView.										
BM3	TWiki AllowWebChange	TestUser1	-	-	-	SIM_UNCLASSIFIED	SIM_UNCLASSIFIED	Write equal	Pass	Fail	Pass
	Test BM3 ensures that a user is able to read wiki page with access granted through AllowWebView.										
BM4	TWiki DenyWebRename	TestUser1	TestUser1	-	-	SIM_UNCLASSIFIED	SIM_UNCLASSIFIED	Write equal	Fail	Fail	Pass
	Test BM4 ensures that a user is not able to read wiki page that has access denied through DenyWebView.										
BM5	TWiki View DAC Not overriding OS DAC	-	-	TestUser1, TestUser2	-	SIM_UNCLASSIFIED	SIM_UNCLASSIFIED	Read equal	Fail	Fail	Fail
	Test BM5 ensures that TWiki View DAC does not overwrite OS DAC (file permission bit set to 000)										
BM6	TWiki Rename DAC Not overriding OS DAC	-	-	TestUser1, TestUser2	-	SIM_UNCLASSIFIED	SIM_UNCLASSIFIED	Write equal	Fail	Fail	Fail
	Test BM6 ensures that TWiki Rename DAC does not overwrite OS DAC (directory permission bit set to 555)										

Table 16. Multilevel XTS TWiki DAC Test.

2. MAC Test Plan – MLS Configuration

MAC tests were performed to ensure that the MAC policies were being enforced correctly. The MAC test cases involve reading and writing object with a security level equal to, above and below the established session level. Table 17 shows the MAC tests that were performed.

Test #	Test Type	TWiki DAC: VIEW & CHANGE Permission Mode Setting				Session Level	Object Level	Operation	Expected Result for Subject: TestUser1
		Allow Web	Deny Web	Allow Topic	Deny Topic				
CM1	Secrecy Read Up	-	-	TestUser1	-	SIM_ UNCLASSIFIED	SIM_SECRET	Read	Fail
	Test CM1 ensures that a user is not able to read wiki page at a secrecy level higher than the level of the currently established session.								
CM2	Secrecy Read Down	-	-	TestUser1	-	SIM_SECRET	SIM_ UNCLASSIFIED	Read	Pass
	Test CM2 ensures that a user is able to read wiki page at a secrecy level lower than the level of the currently established session.								
CM3	Secrecy Read Equal	-	-	TestUser1	-	SIM_ UNCLASSIFIED	SIM_SECRET	Read	Pass
	Test CM3 ensures that a user is able to read wiki page at a secrecy level equal to the level of the currently established session.								
CM4	Secrecy Write Up	-	-	TestUser1	-	SIM_ UNCLASSIFIED	SIM_ CONFIDENTIAL	Write	Fail
	Test CM4 ensures that a user is not able to write to wiki page at a secrecy level higher than the level of the currently established session.								
CM5	Secrecy Write Down	-	-	TestUser1	-	SIM_SECRET	SIM_UNCLASSI FIED	Write	Fail
	Test CM5 ensures that a user is not able to writeto wiki page at a secrecy level lower than the level of the currently established session.								
CM6	Secrecy Write Equal	-	-	TestUser1	-	SIM_ UNCLASSIFIED	SIM_ UNCLASSIFIED	Write	Pass
	Test CM6 ensures that a user is able to write to wiki page at a secrecy level equal to the level of the currently established session.								
CM7	Secrecy Write Equal & Read Up	-	-	TestUser1	-	SIM_ UNCLASSIFIED	SIM_ UNCLASSIFIED	Write & Read	Write Pass Read Fail
	Test CM7 ensures that although a user is able to create a link pointing to wiki page with a secrecy level higher than the level of the currently established session, the user is not able to view or create that page.								

Table 17. Multilevel XTS MAC Tests.

3. Test Results – MLS Configuration

Table 18 shows that the results of the functional tests on the multilevel XTS configuration were the same as those of the Linux server and the single level XTS configuration. This shows that no functionality was lost during the transition to multilevel XTS.

Test #	Test Type	Internet Explorer		FireFox	
		TestUser1	TestUser2	TestUser1	TestUser2
A1	List	pass	pass	pass	pass
A2	Display	pass	pass	pass	pass
A3	Download	pass	pass	pass	pass
A4	Edit	pass	pass	pass	pass
A5	Create	pass	pass	pass	pass
A6	Delete	pass	pass	pass	pass
A7	Upload	pass	pass	pass	pass
A8	Simultaneous Edit	pass	pass	pass	pass
A9	Setting Permission	pass	pass	pass	pass
A10	Configure	pass	pass	pass	pass
A11	Register User	pass	pass	pass	pass

Table 18. Multilevel XTS Functional Test Results.

Table 19 shows the results of the multilevel XTS TWiki DAC testing. The results show that all wiki operations on the clients behaved as expected when constrained by the TWiki DAC policy.

Test #	Test Type	Internet Explorer			FireFox		
		TestUser1	TestUser2	AdminUser	TestUser1	TestUser2	AdminUser
BM1	TWiki DenyTopicView	Fail	Fail	Pass	Fail	Fail	Pass
BM2	TWiki AllowTopicView	Pass	Fail	Pass	Pass	Fail	Pass
BM3	TWiki AllowWebChange	Pass	Fail	Pass	Pass	Fail	Pass
BM4	TWiki DenyWebRename	Fail	Fail	Pass	Fail	Fail	Pass
BM5	TWki View DAC not overriding OS DAC	Fail	Fail	Fail	Fail	Fail	Fail
BM6	TWki Rename DAC not overriding OS DAC	Fail	Fail	Fail	Fail	Fail	Fail

Table 19. Multilevel XTS TWiki DAC Test Results.

Table 20 shows the results of the multilevel level XTS MAC testing. The results show that all wiki operations on the clients behaved as expected when constrained by the MAC policy.

Test #	Test Type	Internet Explorer	FireFox
CM1	Secrecy Read Up	Fail	Fail
CM2	Secrecy Read Down	Pass	Pass
CM3	Secrecy Read Equal	Pass	Pass
CM4	Secrecy Write Up	Fail	Fail
CM5	Secrecy Write Down	Fail	Fail
CM6	Secrecy Write Equal	Pass	Pass
CM7	Secrecy Write Equal & Read Up	Write Pass Read Fail	Write Pass Read Fail

Table 20. Multilevel XTS MAC Test Results.

E. INTEGRATION TEST PLAN

Integration testing was performed to verify that the federated wiki server was able to function correctly in the MYSEA environment, and that the users were able to use the services provided by the two MYSEA servers. The test involved the user switching between the WebDAV services running on one server and the wiki services running on another server and then verifying that those services continued to function correctly for the user. Table 21 shows the two tests that were performed.

Test #	Test Type	Expected Result	Test Objective
D1	Use Wiki on Server X	pass	Ensure that user can only access the wiki from server X
D2	Use WebDAV on Server Y	pass	Ensure that user can access WebDAV on server Y

Table 21. Lists of Integration Test.

1. Test Results

Table 22 shows that the TWiki server can integrate with the existing MYSEA testbed environment as users were able to switch between the WebDAV and wiki services on different servers without losing any functionality. However, users must perform two separate logins as the two servers are not fully federated.

Test #	Test Type	Internet Explorer	FireFox
D1	Use Wiki on Server X	pass	pass
D2	Use WebDAV on Server Y	pass	pass

Table 22. Multilevel XTS Integration Test Results.

F. SUMMARY

This chapter outlined the functional, TWiki DAC, MAC and integration tests performed on three stages of the wiki implementation: Linux, single level XTS, and multilevel XTS. It also summarized the test results, which were consistent with the expected results. The next chapter will discuss the conclusion of the work performed, and suggest possible future work in related areas.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSION AND FUTURE WORK

This chapter states the conclusion of the thesis, discusses a related project and suggests possible future work.

A. CONCLUSION

The goals of this thesis were to determine whether it is possible to design a wiki architecture for a MLS environment, and if so to demonstrate the implementation of such a MLS wiki in the MYSEA testbed. These goals have been successfully demonstrated through the implementation of TWiki on a separate MLS server in the MYSEA testbed. The tests described in Chapter VI confirmed that the MLS-aware wiki server is fully functional and is properly constrained by the underlying MAC and DAC policies enforced by the STOP OS.

B. RELATED WORK

Galois Inc. has a cross-domain solution called the Trusted Services Engine (TSE) that implements WebDAV on a MILS separation kernel to provide WebDAV functionality similar to the MYSEA WebDAV server [18]. Galois Inc. is also working on wiki services for their cross-domain solution. The details of their cross-domain wiki services are not publicly available, but it is understood that Galois plans to construct a wiki with features such as read down and cross-domain merge [19]. The wiki server in MYSEA as presented in this thesis supports read down capability, but a cross domain fusion capability is left for future work.

C. FUTURE WORK

The following issues of arose from this study and warrant additional further work.

1. Wiki Password Synchronization

TWiki maintains its own database of user IDs and passwords. Currently, there is no synchronization between the XTS system's IDs and passwords, and TWiki's IDs and passwords, and users are required to maintain two separate sets of IDs and passwords. A single sign-on solution would provide a more usable interface for the users in terms of user ID and password maintenance and thus enhance the user experience.

2. Cross Domain Content Merge

Cross domain content merging would enable users to login at different session levels to see relevant contents up to their established level, instead of having to browse to individual directories at different security levels to view the content at each level separately. Such a capability would present users with a more unified view of related contents within the wiki, thus enhancing both the user experience and user productivity.

3. Removal of SMTP and IMAP Services on Wiki Server

The SMTP and IMAP services running on the MYSEA server allow mail applications to establish connections to the server. With the additional wiki server, user will have multiple sets of mailboxes, one for each server. This creates usability issues as users must maintain and manage separate sets of mailboxes. The SMTP and IMAP services could be removed by reconfiguration of the MYSEA secure session services to further enhanced the security of the wiki server.

APPENDIX A: INSTALLATION PROCEDURES

This appendix outlines the installation procedures for the TWiki 4.1.2 wiki engine on RedHat Linux 8.0, single level XTS, as well as multilevel XTS.

The following instructions make reference to the Secure Attention Key (SAK) of the XTS-400 machine, which is invoked by simultaneously pressing the "alt" and the "SysRq" (a.k.a. Print Screen) keys. The SAK establishes a trusted path and allows trusted commands to be executed. Two trusted commands are used frequently and are therefore mentioned here: the `sl` command and the `fsm` command. The `sl` command is used to set the security level of the user session, and the `fsm` command is used to display and change the MAC and DAC properties of the files and directories.

A. INSTALLATION AND TEST TOPOLOGY

The test setup consists of two XTS-400 servers running STOP OS 6.3 and two Windows XP clients, connected via a dedicated network hub, as shown in Figure 12. The Windows XP client has Internet Explorer 6 browser installed which is used as the client software for accessing the TWiki engine. Special TPE software written to run on a desktop is also installed on the client to allow user to login to XTS-400 server.

The MYSEA wiki installation CD is needed to perform the installation procedures.

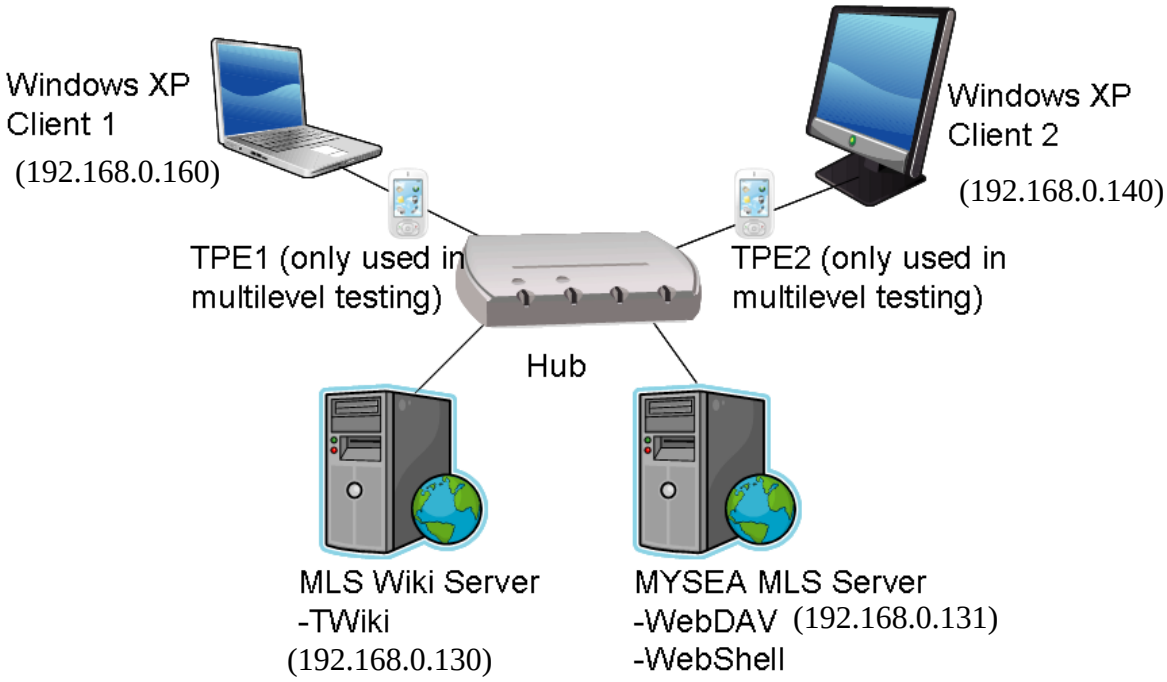

Figure 12. Test Setup Network Topology.

B. LINUX INSTALLATION

The procedures outlined in this section are to be performed as a *root* user on a machine running RedHat 8.0 Linux. RedHat 8.0 is used because the XTS-400 STOP 6.3 operating system is binary compatible to it. These procedures will copy the source code from the wiki installation CD, install and configure the Apache web server and the TWiki 4.1.2 wiki engine on the server. In this section, a base installation of *perl* scripting engine is assumed.

Step 1. Copy the Apache web server, the required Comprehensive Perl Archive Network (CPAN) modules, and the TWiki wiki engine from the MYSEA wiki installation CD:

```
mount /dev/cdrom /mnt/cdrom
cd /root
cp /mnt/cdrom/apache_1.3.34.tar.gz ./
```

```
cp /mnt/cdrom/TWiki-4.1.2.tgz ./
cp /mnt/cdrom/CPAN/perl-Error-0.15-2.0.rh8.dag.noarch.rpm ./
cp /mnt/cdrom/CPAN/perl-FreezeThaw-0.43-
0.dag.rh80.noarch.rpm ./
cp /mnt/cdrom/CPAN/perl-Time-modules-2003.1126-
     1.0.rh9.rf.noarch.rpm ./
cp /mnt/cdrom/CPAN/perl-DBI-1.40-5.i386.rpm ./
cp /mnt/cdrom/CPAN/perl-DB_File-1.804-55.i386.rpm ./
cp /mnt/cdrom/CPAN/perl-CGI-Session-4.00_08-
     1.0.rh9.rf.noarch.rpm ./
cp /mnt/cdrom/CPAN/perl-CGI-2.81-88.3.i386.rpm ./
cp /mnt/cdrom/CPAN/libpng10-1.0.13-5.i386.rpm
 ./
cp /mnt/cdrom/CPAN/gd-1.8.3-4.i386.rpm ./
cp /mnt/cdrom/CPAN/perl-GD-1.41-0.rh80.dag.i386.rpm ./
cp /mnt/cdrom/CPAN/perl-HTML-Tagset-3.10-1.rh9.rf.noarch.rpm
     ./
cp /mnt/cdrom/CPAN/perl-HTML-Parser-3.55-1.rh9.rf.i386.rpm
     ./
cp /mnt/cdrom/CPAN/perl-HTML-Tree-3.17-0.dag.rh80.noarch.rpm
     ./
```

Step 2. Install the required CPAN modules:

```
rpm –ivh perl-Error-0.15-2.0.rh8.dag.noarch.rpm
rpm –ivh perl-FreezeThaw-0.43-0.dag.rh80.noarch.rpm
rpm –ivh perl-Time-modules-2003.1126-1.0.rh9.rf.noarch.rpm
rpm –ivh perl-DBI-1.40-5.i386.rpm
rpm –ivh perl-DB_File-1.804-55.i386.rpm
rpm –ivh perl-CGI-Session-4.00_08-1.0.rh9.rf.noarch.rpm
rpm –ivh perl-CGI-2.81-88.3.i386.rpm
rpm –ivh libpng10-1.0.13-5.i386.rpm
rpm –ivh gd-1.8.3-4.i386.rpm
rpm –ivh perl-GD-1.41-0.rh80.dag.i386.rpm
rpm –ivh perl-HTML-Tagset-3.10-1.rh9.rf.noarch.rpm
rpm –ivh perl-HTML-Parser-3.55-1.rh9.rf.i386.rpm
rpm –ivh perl-HTML-Tree-3.17-0.dag.rh80.noarch.rpm
```

Step 3. Unpack, build and install the Apache web server:

```
tar –zxvf apache_1.3.34.tar.gz
cd /root/apache_1.3.34
./configure --prefix=/www
make
make install
```

Step 4. Save and edit the default Apache configuration:

```
cd /www/conf
cp httpd.conf httpd.conf.bak
vi httpd.conf
```

Step 5. Append the following lines to the “http.conf” file:

```
<Directory /www/htdocs/twiki/bin>
     Options ExecCGI FollowSymLinks
     Order allow,deny
     Allow from all
     SetHandler cgi-script
</Directory>
```

Step 6: Change the user and group of Apache daemon in httpd.conf, and save the file:

```
User apache
Group apache
```

Step 7: Create the apache user and group:

```
adduser apache
```

Step 8. Start the Apache web server:

```
/www/bin/apachectl start
```

Step 9. Unpack and configure TWiki files:

```
mkdir /www/htdocs/twiki
cp /root/TWiki-4.1.2.tgz /www/htdocs/twiki/
cd /www/htdocs/twiki
tar -zxvf TWiki-4.1.2.tgz
cp ./bin/LocalLib.cfg.txt ./bin/LocalLib.cfg
chown –R apache:apache /www/htdocs/twiki
```

Step 10: Edit TWiki LocalLib.cfg library configuration file:

```
cd /www/htdocs/twiki/bin
vi LocalLib.cfg
```

Step 11: Change the path to lib directory of the TWiki installation and save the file:

```
$twikiLibPath = "/www/htdocs/twiki/lib"
```

Step 12: Configure the TWiki engine by starting a web browser and access the TWiki *configure* script through the following URL:

```
http://<server-ip-address>/twiki/bin/configure
```

Click "Save" at the bottom of the page to ignore the warning messages under "General Path Setting". At the next prompt, click save again, this will bring to a "Updating configuration" page with a summary of the configuration changes. Click 'Return to Configuration".

- Under "Security Setups->Authentication->{Login Manager}", select "Template Login".
- Under "Security Setups->Registration-> {Register}{NeedVerification}", uncheck the checkbox.
- Under "Store Settings->{StoreImpl}, select "RCSLite".
- Under "Mail and Proxies->{WebMasterEmail}", enter admin@mlsserver.cisrlabmlstestbed1.com.
- Under "Mail and Proxies->{MailPrograml}", leave the field blank.
- Click "Next" at the bottom of the page.
- Select and confirm a password, and click save, and then click "Return to configuration" on the next page that appears.

Step 13: Verify that the installation is working, by clicking on the link "browse to the TWiki WebHome" in the configuration page of Step 12. A user should then be able to login as TestUser1 and browse to the Main WebHome page by clicking on the "Main Web" link. A page with the header "Welcome to the Main Web" with the text "Congratulations, you have finished the TWiki installation" will be displayed.

Once all these steps are performed, the installation of Apache and TWiki on RedHat Linux has been completed. The tests outlined in Section 2.1 and 2.4 of Appendix B should be performed to ensure that the software is working correctly.

C. SINGLE LEVEL XTS INSTALLATION

The procedures outlined in this section are to be performed as the "admin" user on the XTS-400 machine running standard STOP configuration. These procedures will copy the source code from the wiki installation CD, install and configure the Apache 1.3.34 web server and the TWiki 4.1.2 wiki engine on the server. In this section, perl v5.8.0 or above scripting engine is assumed to be installed on the server.

In a single level XTS-400 environment, all data coming in and out of the server takes on the security level of the network interface card. Therefore, the following instructions must be performed under the session level sl1:il3, which is the security level configured for the network interface card for the single level prototype.

Step 1. Copy the Apache web server, the required CPAN modules, and the TWiki wiki engine from the MYSEA installation CD:

(set security and integrity levels – min:oss)

```
SAK
Enter command?                    run
cd /home/admin
cdtool cp /dev/cdrom /apache_1.3.34.tar.gz
      ./apache_1.3.34.tar.gz
cdtool cp /dev/cdrom /TWiki-4.1.2.tgz ./TWiki-4.1.2.tgz
cdtool cp /dev/cdrom /CPAN/perl-Error-0.15-
      2.0.rh8.dag.noarch.rpm ./perl-Error-0.15-
      2.0.rh8.dag.noarch.rpm
cdtool cp /dev/cdrom /CPAN/perl-FreezeThaw-0.43-
      0.dag.rh80.noarch.rpm ./perl-FreezeThaw-0.43-
      0.dag.rh80.noarch.rpm
cdtool cp /dev/cdrom /CPAN/perl-Time-modules-2003.1126-
      1.0.rh9.rf.noarch.rpm ./perl-Time-modules-2003.1126-
      1.0.rh9.rf.noarch.rpm
cdtool cp /dev/cdrom /CPAN/perl-DBI-1.40-5.i386.rpm ./perl-
      DBI-1.40-5.i386.rpm
cdtool cp /dev/cdrom /CPAN/perl-DB_File-1.804-55.i386.rpm
      ./perl-DB_File-1.804-55.i386.rpm
cdtool cp /dev/cdrom /CPAN/perl-CGI-Session-4.00_08-
      1.0.rh9.rf.noarch.rpm ./ perl-CGI-Session-4.00_08-
      1.0.rh9.rf.noarch.rpm
cdtool cp /dev/cdrom /CPAN/perl-CGI-2.81-88.3.i386.rpm
      ./perl-CGI-2.81-88.3.i386.rpm
```

```
cdtool cp /dev/cdrom /CPAN/libpng10-1.0.13-5.i386.rpm
     ./libpng10-1.0.13-5.i386.rpm
cdtool cp /dev/cdrom /CPAN/gd-1.8.3-4.i386.rpm ./gd-1.8.3-
      4.i386.rpm
cdtool cp /dev/cdrom /CPAN/perl-GD-1.41-0.rh80.dag.i386.rpm
      ./perl-GD-1.41-0.rh80.dag.i386.rpm
cdtool cp /dev/cdrom /CPAN/perl-HTML-Tagset-3.10-
      1.rh9.rf.noarch.rpm ./perl-HTML-Tagset-3.10-
      1.rh9.rf.noarch.rpm
cdtool cp /dev/cdrom /CPAN/perl-HTML-Parser-3.55-
      1.rh9.rf.i386.rpm ./perl-HTML-Parser-3.55-
      1.rh9.rf.i386.rpm
cdtool cp /dev/cdrom /CPAN/perl-HTML-Tree-3.17-
      0.dag.rh80.noarch.rpm ./perl-HTML-Tree-3.17-
      0.dag.rh80.noarch.rpm
```

Step 2. Change the security level of the TWiki and Apache files to sl1:il3:

SAK	
Enter command?	`fsm`
Enter request?	`change`
Enter pathname?	`/home/admin/TWiki-4.1.2.tgz`
Modify access level?	`yes`
Enter new security level?	`sl1`
Enter new integrity level?	`il3`
Is the level correct?	`yes`
Enter new owner name?	`admin`
Enter new group name?	`stop`
Modify discretionary access?	`no`
Display the object?	`no`
Okay to change?	`yes`

Repeat the above steps for the file: apache_1.3.34.tar.gz, entering the pathname /home/admin/apache_1.3.34.tar.gz .

Step 3. Install the required CPAN modules:

(set security and integrity levels – min:oss)

SAK	
Enter command?	`run`

```
rpm –ivh perl-Error-0.15-2.0.rh8.dag.noarch.rpm
```

```
rpm –ivh perl-FreezeThaw-0.43-0.dag.rh80.noarch.rpm
rpm –ivh perl-Time-modules-2003.1126-1.0.rh9.rf.noarch.rpm
rpm –ivh perl-DBI-1.40-5.i386.rpm
rpm –ivh perl-DB_File-1.804-55.i386.rpm
rpm –ivh perl-CGI-Session-4.00_08-1.0.rh9.rf.noarch.rpm
rpm –ivh perl-CGI-2.81-88.3.i386.rpm
rpm –ivh libpng10-1.0.13-5.i386.rpm
rpm –ivh gd-1.8.3-4.i386.rpm
rpm –ivh perl-GD-1.41-0.rh80.dag.i386.rpm
rpm –ivh perl-HTML-Tagset-3.10-1.rh9.rf.noarch.rpm
rpm –ivh perl-HTML-Parser-3.55-1.rh9.rf.i386.rpm
rpm –ivh perl-HTML-Tree-3.17-0.dag.rh80.noarch.rpm
```

Step 4. Create Apache directory for sl1:il3:

(set security and integrity level – min:oss)

SAK	
Enter command?	`fsm`
Enter request	`mkdir`
Enter the directory to create	`/home/admin/apache`
Should this be a deflection directory	`No`
[fsm:Main]	
Enter request	`change`
Enter pathname?	`/home/admin/apache`
Modify access level?	`yes`
Enter new security level?	`sl1`
Enter new integrity level?	`il3`
Is the level correct?	`Yes`
Modify discretionary access?	`Yes`
Enter new owner name?	`admin`
Enter new group name?	`mysea`
Enter object mode for owner?	`rwx`
Enter object mode for group?	`rx`
Enter object mode for others?	`rx`
Display the object?	`no`
Okay to change?	`yes`

Step 5. Unpack, build and install the Apache web server:

(set security and integrity levels – sl1:il3)

SAK
Enter command? run

```
cd /home/admin/
cp apache_1.3.34.tar.gz ./apache
cd ./apache
tar –zxvf apache_1.3.34.tar.gz
cd /home/admin/apache/apache_1.3.34
./configure --prefix=/home/admin/apache/
make
make install
```

Step 6. Save and edit the default Apache configuration:

```
cd /home/admin/apache/conf
cp httpd.conf httpd.conf.bak
vi httpd.conf
```

Step 7. Append the following lines to the “http.conf” file and save the file:

```
<Directory /home/admin/apache/htdocs/twiki/bin>
     Options ExecCGI FollowSymLinks
     Order allow,deny
     Allow from all
     SetHandler cgi-script
</Directory>
```

Step 8. Start the Apache web server:

```
/home/admin/apache/bin/apachectl start
```

Step 9. Unpack and configure TWiki files:

```
mkdir /home/admin/apache/htdocs/twiki
cp /home/admin/TWiki-4.1.2.tgz /home/admin/apache/htdocs/twiki
cd /home/admin/apache/htdocs/twiki
tar -zxvf TWiki-4.1.2.tgz
cp ./bin/LocalLib.cfg.txt ./bin/LocalLib.cfg
```

Step 10: Edit TWiki LocalLib.cfg library configuration file:

```
cd /home/admin/apache/htdocs/twiki/bin
vi LocalLib.cfg
```

Step 11: Change the path to lib directory of the TWiki installation and save the file:

```
$twikiLibPath = "/home/admin/apache/htdocs/twiki/lib"
```

Step 12: Create twiki temporary directory:

```
cd /tmp
mkdir twiki
```

Step 13: Configure the TWiki engine by starting a web browser and access the TWiki *configure* script through the following URL:

`http://<server-ip-address>/twiki/bin/configure`

Click **"Save"** at the bottom of the page to ignore the warning messages under "General Path Setting". At the next prompt, click save again, this will bring to a "Updating configuration" page with a summary of the configuration changes. Click 'Return to Configuration".

- Under "Security Setups->Authentication->{Login Manager}", select "Template Login".
- Under "Security Setups->Registration-> {Register}{NeedVerification}", uncheck the checkbox.
- Under "Store Settings->{StoreImpl}, select "RCSLite".
- Under "Mail and Proxies->{WebMasterEmail}", enter admin@mlsserver.cisrlabmlstestbed1.com.
- Under "Mail and Proxies->{MailPrograml}", leave the field blank.
- Click "Next" at the bottom of the page.
- Select and confirm a password, and click save, and then click "Return to configuration" on the next page that appears.

Step 14: Verify that the installation is working, by clicking on the link "browse to the TWiki WebHome" in the configuration page of Step 13. A user should then be able to login as TestUser1 and browse to the Main WebHome page by clicking on the "Main Web" link. A page with the header "Welcome to the Main Web" with the text "Congratulations, you have finished the TWiki installation" will be displayed.

Once all these steps are performed, the installation of Apache and TWiki on the single level XTS has been completed. The tests outlined in Section 2.2, 2.5 and 2.7 of Appendix B should be performed to ensure that the software is working correctly.

D. MULTI LEVEL XTS INSTALLATION

The procedures outlined in this section are to be performed as the "admin" user on the XTS-400 machine. These procedures will copy the source code from the wiki installation CD, install and configure the TWiki 4.1.2 wiki engine on the server. In this section, MYSEA version 2.0, Apache 1.3.34 and perl 5.8.0 or above are assumed to be installed.

Step 1. Copy the TWiki wiki engine and CPAN modules from the installation CD:

(set security and integrity level - min:oss)

SAK
Enter command? run

```
cd /home/admin
cdtool cp /dev/cdrom /TWiki-4.1.2-MYSEA.tgz ./TWiki-4.1.2-
      MYSEA.tgz
cdtool cp /dev/cdrom /CPAN/perl-Error-0.15-
      2.0.rh8.dag.noarch.rpm ./perl-Error-0.15-
      2.0.rh8.dag.noarch.rpm
cdtool cp /dev/cdrom /CPAN/perl-FreezeThaw-0.43-
      0.dag.rh80.noarch.rpm ./perl-FreezeThaw-0.43-
      0.dag.rh80.noarch.rpm
cdtool cp /dev/cdrom /CPAN/perl-Time-modules-2003.1126-
      1.0.rh9.rf.noarch.rpm ./perl-Time-modules-2003.1126-
      1.0.rh9.rf.noarch.rpm
cdtool cp /dev/cdrom /CPAN/perl-DBI-1.40-5.i386.rpm ./perl-
      DBI-1.40-5.i386.rpm
cdtool cp /dev/cdrom /CPAN/perl-DB_File-1.804-55.i386.rpm
      ./perl-DB_File-1.804-55.i386.rpm
cdtool cp /dev/cdrom /CPAN/perl-CGI-Session-4.00_08-
      1.0.rh9.rf.noarch.rpm ./ perl-CGI-Session-4.00_08-
      1.0.rh9.rf.noarch.rpm
```

```
cdtool cp /dev/cdrom /CPAN/perl-CGI-2.81-88.3.i386.rpm
       ./perl-CGI-2.81-88.3.i386.rpm
cdtool cp /dev/cdrom /CPAN/libpng10-1.0.13-5.i386.rpm
     ./libpng10-1.0.13-5.i386.rpm
cdtool cp /dev/cdrom /CPAN/gd-1.8.3-4.i386.rpm ./gd-1.8.3-
       4.i386.rpm
cdtool cp /dev/cdrom /CPAN/perl-GD-1.41-0.rh80.dag.i386.rpm
       ./perl-GD-1.41-0.rh80.dag.i386.rpm
cdtool cp /dev/cdrom /CPAN/perl-HTML-Tagset-3.10-
       1.rh9.rf.noarch.rpm ./perl-HTML-Tagset-3.10-
       1.rh9.rf.noarch.rpm
cdtool cp /dev/cdrom /CPAN/perl-HTML-Parser-3.55-
       1.rh9.rf.i386.rpm ./perl-HTML-Parser-3.55-
       1.rh9.rf.i386.rpm
cdtool cp /dev/cdrom /CPAN/perl-HTML-Tree-3.17-
       0.dag.rh80.noarch.rpm ./perl-HTML-Tree-3.17-
       0.dag.rh80.noarch.rpm
```

Step 2. Change the security level of the TWiki files to sl1:il0:

```
SAK
Enter command?              fsm
Enter request?              change
Enter pathname?             /home/admin/TWiki-4.1.2-MYSEA.tgz
Modify access level?        yes
Enter new security level?   sl1
Enter new integrity level?  il0
Is the level correct?       Yes
Modify discretionary access? no
Display the object?         no
Okay to change?             yes
```

Step 3. Install the required CPAN modules:

(set security and integrity levels – min:oss)

```
SAK
Enter command?          run

rpm –ivh perl-Error-0.15-2.0.rh8.dag.noarch.rpm
rpm –ivh perl-FreezeThaw-0.43-0.dag.rh80.noarch.rpm
rpm –ivh perl-Time-modules-2003.1126-1.0.rh9.rf.noarch.rpm
rpm –ivh perl-DBI-1.40-5.i386.rpm
rpm –ivh perl-DB_File-1.804-55.i386.rpm
```

```
rpm –ivh perl-CGI-Session-4.00_08-1.0.rh9.rf.noarch.rpm
rpm –ivh perl-CGI-2.81-88.3.i386.rpm
rpm –ivh libpng10-1.0.13-5.i386.rpm
rpm –ivh gd-1.8.3-4.i386.rpm
rpm –ivh perl-GD-1.41-0.rh80.dag.i386.rpm
rpm –ivh perl-HTML-Tagset-3.10-1.rh9.rf.noarch.rpm
rpm –ivh perl-HTML-Parser-3.55-1.rh9.rf.i386.rpm
rpm –ivh perl-HTML-Tree-3.17-0.dag.rh80.noarch.rpm
```

Step 4. Skip to Step 8 if WebDAV if is not installed on the server. Edit the Apache install configuration:

(set security and integrity level – sl0:il3)

```
cd /usr/local/mysea/apache/src
vi Configuration
```

Step 5. Comment out the following text at the end of the "Configuration" file to remove the WebDAV module:

```
# AddModule modules/dav/libdav.a
```

Step 6: Build Apache:

```
./Configure
make
```

Step 7: Replace Apache httpd binary file:

```
cd /usr/local/mysea/bin
mv httpd httpd_webdav
cp /usr/local/mysea/apache/src/httpd ./
```

Step 8. Save and edit the default Apache configuration:

(set security and integrity level – sl0:il3)

SAK
Enter command? `run`

```
cd /home/http/conf
cp httpd.conf httpd.conf.bak
vi httpd.conf
```

Step 9. Append the following lines to the “http.conf” file:

```
<Directory /home/http/htdocs/twiki/bin>
     Options ExecCGI FollowSymLinks
     Order allow,deny
     Allow from all
     SetHandler cgi-script
</Directory>
```

Step 10 : Skip to Step 11 if WebDAV is not installed on the server. Comment out the following text from the “httpd.conf” file:

```
# DAVLockDB /usr/local/apachel3/var/DAVLock
# <Location /dav>
#    DAV on
# </Location>
```

Step 11: Skip to Step 12 if WebShell is not installed on the server. Delete the WebShell cgi script:

```
cd /home/http/cgi-bin
rm WebShell.cgi
```

Step 12. Create a *twiki* directory for sl1:il0 to unpack and configure TWiki files:

(set security and integrity level – sl0:il3)

SAK

Enter command?	`fsm`
Enter request	`mkdir`
Enter the directory to create	`/home/http/htdocs/twiki`
Should this be a deflection directory	`No`
[fsm:Main]	
Enter request	`change`
Enter pathname?	`/home/http/htdocs/twiki`
Modify access level?	`yes`
Enter new security level?	`sl1`

Enter new integrity level? il0
Is the level correct? Yes
Modify discretionary access? Yes
Enter new owner name? admin
Enter new group name? other
Enter object mode for owner? rwx
Enter object mode for group? rwx
Enter object mode for others? rx
Display the object? no
Okay to change? yes

(set security and integrity level – sl1:il0)

SAK
Enter command? run

```
cp /home/admin/TWiki-4.1.2-MYSEA.tgz /home/http/htdocs/twiki
cd /home/http/htdocs/twiki
tar -zxvf TWiki-4.1.2-MYSEA.tgz
cd ..
chown –R admin:other /home/http/htdocs/twiki
```

Step 13: Rename default Trash directory:

```
cd /home/http/htdocs/twiki/data
mv Trash Trash_org
```

Step 14: Re-create Trash directory as deflection directory:

(set security and integrity level – sl1:il0)

SAK

Enter command? fsm
Enter request? mkdir
Enter pathname? /home/http/htdocs/twiki/data/Trash
Is it a deflection dir? yes
Enter directory mode for owner? rwx
Enter directory mode for group? rwx
Enter directory mode for others? rx

Step 15: Create content directory at the SIM_UNCLASSIFIED security level: (set security and integrity level – sl1:il0)

```
SAK
Enter command?                          fsm
Enter request                           mkdir
Enter the directory to create
        /home/http/htdocs/twiki/data/SIM_UNCLASSIFIED
Should this be a deflection directory   No
[fsm:Main]
Enter request                           change
Enter pathname?
        /home/http/htdocs/twiki/data/SIM_UNCLASSIFIED
Modify access level?                    yes
Enter new security level?               sl1
Enter new integrity level?              il0
Is the level correct?                   Yes
Modify discretionary access?            Yes
Enter new owner name?                   admin
Enter new group name?                   other
Enter object mode for owner?            rwx
Enter object mode for group?            rwx
Enter object mode for others?           rx
Display the object?                     no
Okay to change?                         yes
[fsm:Main]
Enter request                           mkdir
Enter the directory to create
        /home/http/htdocs/twiki/pub/SIM_UNCLASSIFIED
Should this be a deflection directory   No
[fsm:Main]
Enter request                           change
Enter pathname?
        /home/http/htdocs/twiki/pub/SIM_UNCLASSIFIED
Modify access level?                    yes
Enter new security level?               sl1
Enter new integrity level?              il0
Is the level correct?                   Yes
Modify discretionary access?            Yes
Enter new owner name?                   admin
Enter new group name?                   other
Enter object mode for owner?            rwx
Enter object mode for group?            rwx
```

Enter object mode for others? rx
Display the object? no
Okay to change? yes

Step 16: Populate content directory at the SIM_UNCLASSIFIED security level:

(set security and integrity level – sl1:il0)

SAK
Enter command? run

```
cd /home/http/htdocs/twiki/data/SIM_UNCLASSIFIED
cp –r /home/http/htdocs/twiki/data_testing/SIM_UNCLASSIFIED/* ./
cd ..
cp –r /home/http/htdocs/twiki/data_testing/Main/ ./
chown –R admin:other SIM_UNCLASSIFIED
chown –R admin:other Main
cd /home/http/htdocs/twiki/pub/SIM_UNCLASSIFIED
cp –r /home/http/htdocs/twiki/pub_testing/SIM_UNCLASSIFIED/* ./
cd ..
chown –R admin:other SIM_UNCLASSIFIED
```

Step 17: Populate Trash directory at each security level:

(set security and integrity level – sl1:il0)

```
cd /home/http/htdocs/twiki/data
chmod 775 Trash
cd Trash
cp ../Trash_org/* ./
chmod 664 *
cd ..
chown –R admin:other Trash
```

Step 18: Create twiki temporary directory at the SIM_UNCLASSIFIED security level:

(set security and integrity level – sl1:il0)

```
cd /tmp
mkdir twiki
chmod 775 twiki
chown admin:other twiki
```

Repeat the Step 15 to 18 for the following classification level while logged in at the appropriate security and integrity level:

SIM_SECRET sl5 (sc1) : il0

Step 19: Restore files to be used for testing from tape:

(set security and integrity level – max:max)

```
SAK
Enter command?   frestore
   /dev/tape1    for input source
   <CR>          for volume name
(when instructed, load tape, write protected 'slide
open')
   restore       for enter request
   n    for replace newer objects with older
   n    for display pathnames
   n    for restore access and modify times
   y    for restore owner/group of restored objects
   <CR> for relative pathname selection
   <CR> for destination pathname (default should be
"/home/http/htdocs/twiki/data"
   Exit for Enter request
   (remove tape from tape drive after it is ejected)
```

Step 20: Copy the SIM_UNCLASSIFIED directory under the pub directory restored in Step 19 to /home/http/htdocs/twiki/pub:

(set security and integrity level – sl1:il0)

```
cp –rp /home/http/htdocs/twiki/data/pub/SIM_UNCLASSIFIED
/home/http/htdocs/twiki/pub
```

Step 21: Login to the XTS server through TPE or TCBE at the SIM_UNCLASSIFIED level (sl1:il0). Configure the TWiki engine by starting a web browser and access the TWiki *configure* script through the following URL:

```
http://<server-ip-address>/twiki/bin/configure
```

- Under "Security Setups->Authentication->{Login Manager}", select "Template Login".
- Under "Security Setups->Registration-> {Register}{NeedVerification}", uncheck the checkbox.
- Under "Store Settings->{StoreImpl}", select "RCSLite".

- Under “Store Settings->{RCS}{dirPermission}”, enter “0775”
- Under “Store Settings->{RCS}{filePermission}”, enter “0664”
- Under “Mail and Proxies->{WebMasterEmail}”, enter admin@mlsserver.cisrlabmlstestbed1.com.
- Under “Mail and Proxies->{MailPrograml}”, leave the field blank.
- Click “Next” at the bottom of the page.
- Key in “password” as the password, and click save, and then click “Return to configuration” on the next page that appears.

Step 22: Verify that the installation is working, by clicking on the link “browse to the TWiki WebHome” in the configuration page. A user should then be able to login as TestUser1 and browse to the Main WebHome page by clicking on the “Main Web” link. A page with the header “Welcome to the Main Web” with the text “Congratulations, you have finished the TWiki installation” will be displayed.

Once all these steps are performed, the installation of Apache and TWiki on the multi level XTS has been completed. The tests outlined in Section 2.3, 2.6 and 2.8 of Appendix B should be performed to ensure that the software is working correctly.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: TEST PROCEDURES

A. DESCRIPTION

This TWiki Test Procedure is intended to implement the TWiki Test Plan. This procedure will provide details on testing covered for acceptance of TWiki integration into MYSEA Testbed.

B. TEST DETAILS

1. Test Setups

TWiki Test setup includes the two XTS servers, a Windows clients with Internet Explorer 6.0 browser and a Linux client with FireFox 2.0 browser

Two STOP user accounts are used: *mdemo1* and *mdemo2*.
Three TWiki user accounts are used: *AdminUser*, *TestUser1* and *TestUser2*.
The *mdemo1* user holds the *TestUser1* TWiki account and the *AdminUser* TWiki account, and the *mdemo2* user holds the *TestUser2* TWiki account.

The following tables show the TWiki DAC and OS MAC setting of the test webs and topics. (legend: AU: AdminUser, TU1: TestUser1, TU1,2: TestUser1 & TestUser2,
AT: AllowTopic, DT: DenyTopic, AW: AllowWeb, DW: DenyWeb, N.A.: not applicable, a dash "-" : not set)

TWiki Web	Twiki Topic	Twiki DAC												OS MAC (Obj Label, for XTS only)
		View				Change				Rename				
		AW	DW	AT	DT	AW	DW	AT	DT	AW	DW	AT	DT	
Test_Functional	A1FunctionalTest List	-	-	-	-	-	-	-	-	-	-	-	-	sl1:il3 (single level) sl0:il0 (multilevel)
Test_Functional	A2FunctionalTest Display1	-	-	-	-	-	-	-	-	-	-	-	-	sl1:il3 (single level) sl0:il0 (multilevel)
Test_Functional	A2FunctionalTest Display2	-	-	-	-	-	-	-	-	-	-	-	-	sl1:il3 (single level) sl0:il0 (multilevel)
Test_Functional	A3FunctionalTest Download	-	-	-	-	-	-	-	-	-	-	-	-	sl1:il3 (single level) sl0:il0 (multilevel)
Test_Functional	A4FunctionalTestEdit1	-	-	-	-	-	-	-	-	-	-	-	-	sl1:il3 (single level) sl0:il0 (multilevel)
Test_Functional	A4FunctionalTestEdit2	-	-	-	-	-	-	-	-	-	-	-	-	sl1:il3 (single level) sl0:il0 (multilevel)
Test_Functional	A5FunctionalTestCreate	-	-	-	-	-	-	-	-	-	-	-	-	sl1:il3 (single level) sl0:il0 (multilevel)
Test_Functional	A6FunctionalTestDelete	-	-	-	-	-	-	-	-	-	-	-	-	sl1:il3 (single level) sl0:il0 (multilevel)
Test_Functional	A7FunctionalTestUpload	-	-	-	-	-	-	-	-	-	-	-	-	sl1:il3 (single level) sl0:il0 (multilevel)
Test_Functional	A8FunctionalTestSimultaneousEdit	-	-	-	-	-	-	-	-	-	-	-	-	sl1:il3 (single level) sl0:il0 (multilevel)
Test_Functional	A9FunctionalTestSetPermission	-	-	-	-	-	-	-	-	-	-	-	-	sl1:il3 (single level) sl0:il0 (multilevel)
Test_Functional	A10FunctionalTest Configure	-	-	-	-	-	-	-	-	-	-	-	-	sl1:il3 (single level) sl0:il0 (multilevel)
Test_Functional	A11FunctionalTest RegisterUser	-	-	-	-	-	-	-	-	-	-	-	-	sl1:il3 (single level) sl0:il0 (multilevel)

Table 23. Functional Test Permission Settings

TWiki Web	Twiki Topic	Twiki DAC												OS MAC (Obj Label, for XTS only)
		View				Change				Rename				
		AW	DW	AT	DT	AW	DW	AT	DT	AW	DW	AT	DT	
Test_DAC_Linux/BL1	TestTopic	-	-	TU1	TU1	-	-	-	-	-	-	-	-	N.A.
Test_DAC_Linux/BL2	TestTopic	-	TU1	TU1	-	-	-	-	-	-	-	-	-	N.A.
Test_DAC_Linux/BL3	TestTopic	-	-	-	-	TU1	-	-	-	-	-	-	-	N.A.
Test_DAC_Linux/BL4	TestTopic	-	-	-	-	-	-	-	-	TU1	TU1	-	-	N.A.
Test_DAC_Linux/BL5	TestTopic	-	-	TU1,2	-	-	-	TU1,2	-	-	-	TU1,2	-	N.A.
Test_DAC_Linux/BL6	TestTopic		-	TU1,2	-	-	-	TU1,2	-	-	-	TU1,2	-	N.A.
Test_DAC_Linux/BL7	TestTopic	-	-	TU1,2	-	-	-	TU1,2	-	-	-	TU1,2	-	N.A.
Test_DAC_SingleLevel/BS1	TestTopic	-	-	TU1	TU1	-	-	-	-	-	-	-	-	sl1:il3
Test_DAC_SingleLevel/BS2	TestTopic	-	TU1	TU1	-	-	-	-	-	-	-	-	-	sl1:il3
Test_DAC_SingleLevel/BS3	TestTopic	-	-	-	-	TU1	-	-	-	-	-	-	-	sl1:il3
Test_DAC_SingleLevel/BS4	TestTopic	-	-	-	-	-	-	-	-	TU1	TU1	-	-	sl1:il3
Test_DAC_SingleLevel/BS5	TestTopic	-	-	TU1,2	-	-	-	TU1,2	-	-	-	TU1,2	-	sl1:il3
Test_DAC_SingleLevel/BS6	TestTopic		-	TU1,2	-	-	-	TU1,2	-	-	-	TU1,2	-	sl1:il3
Test_DAC_MultiLevel/BM1	TestTopic	-	-	TU1	TU1	-	-	-	-	-	-	-	-	SIM_UNCLASSIFIED
Test_DAC_MultiLevel/BM2	TestTopic	-	TU1	TU1	-	-	-	-	-	-	-	-	-	SIM_UNCLASSIFIED
Test_DAC_MultiLevel/BM3	TestTopic	-	-	-	-	TU1	-	-	-	-	-	-	-	SIM_UNCLASSIFIED
Test_DAC_MultiLevel/BM4	TestTopic	-	-	-	-	-	-	-	-	TU1	TU1	-	-	SIM_UNCLASSIFIED
Test_DAC_MultiLevel/BM5	TestTopic	-	-	TU1,2	-	-	-	TU1,2	-	-	-	TU1,2	-	SIM_UNCLASSIFIED
Test_DAC_MultiLevel/BM6	TestTopic		-	TU1,2	-	-	-	TU1,2	-	-	-	TU1,2	-	SIM_UNCLASSIFIED

Table 24. DAC Test Permission Settings.

TWiki Web	Twiki Topic	Twiki DAC												OS MAC (Obj Label, for XTS only)
		View				Change				Rename				
		AW	DW	AT	DT	AW	DW	AT	DT	AW	DW	AT	DT	
Test_MAC_SingleLevel/CS1	TestTopic	-	-	TU1	-	-	-	TU1	-	-	-	TU1	-	sl2:il3
Test_MAC_SingleLevel/CS2	TestTopic	-	-	TU1	-	-	-	TU1	-	-	-	TU1	-	sl0:il3
Test_MAC_SingleLevel/CS3	TestTopic	-	-	TU1	-	-	-	TU1	-	-	-	TU1	-	sl1:il3
Test_MAC_SingleLevel/CS4	TestTopic	-	-	TU1	-	-	-	TU1	-	-	-	TU1	-	sl2:il3
Test_MAC_SingleLevel/CS5	TestTopic	-	-	TU1	-	-	-	TU1	-	-	-	TU1	-	sl0:il3
Test_MAC_SingleLevel/CS6	TestTopic	-	-	TU1	-	-	-	TU1	-	-	-	TU1	-	sl1:il3
Test_MAC_MultiLevel/CM1	TestTopic	-	-	TU1	-	-	-	TU1	-	-	-	TU1	-	SIM_SECRET
Test_MAC_MultiLevel/CM2	TestTopic	-	-	TU1	-	-	-	TU1	-	-	-	TU1	-	SIM_UNCLASSIFIED
Test_MAC_MultiLevel/CM3	TestTopic	-	-	TU1	-	-	-	TU1	-	-	-	TU1	-	SIM_UNCLASSIFIED
Test_MAC_MultiLevel/CM4	TestTopic	-	-	TU1	-	-	-	TU1	-	-	-	TU1	-	SIM_SECRET
Test_MAC_MultiLevel/CM5	TestTopic	-	-	TU1	-	-	-	TU1	-	-	-	TU1	-	SIM_UNCLASSIFIED
Test_MAC_MultiLevel/CM6	TestTopic	-	-	TU1	-	-	-	TU1	-	-	-	TU1	-	SIM_UNCLASSIFIED
Test_MAC_MultiLevel/CM7	TestTopic	-	-	TU1	-	-	-	TU1	-	-	-	TU1	-	SIM_UNCLASSIFIED
Test_Integration	TestTopic	-	-	-	-	-	-	-	-	-	-	-	-	SIM_UNCLASSIFIED

Table 25. MAC Test and Integration Test Permission Settings.

2. Test Procedures

2.1. Functional Tests (Linux)

These tests verify that all TWiki functionalities are behaving correctly.

2.1.1. List Wiki Web/Pages Test Instruction (A1)

1. In a new browser window, connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: "A1FunctionalTestList".
5. Expecting a listing of Functional Test wiki pages A1 to A11.
6. Close the browser window.
7. Repeat the above steps using a second client and login to TWiki as TestUser2.
8. Expecting a listing of Functional Test wiki pages A1 to A11
9. Close the browser window.
10. Annotate completion by initialing box to the left.

2.1.2. Display Wiki Pages Test Instruction (A2)

1. In a new browser window, connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: "A2FunctionalTestDisplay1".
5. Expecting a display of the content of the wiki page: "This is Display Test 1".
6. Close the browser window.
7. Repeat the steps 1 to 3 using a second client and login to TWiki as TestUser2.
8. Click on wiki page: "A2FunctionalTestDisplay2".
9. Expecting a display of the content of the wiki page: "This is Display Test 2".
10. Close the browser window.
11. Annotate completion by initialing box to the left.

2.1.3. Download File Test Instruction (A3)

1. In a new browser window, connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar

4. Click on wiki page: “A3FunctionalTestDownload”.
5. Download a file: **right click on the file “Functional_Test_Download_File1.txt”, select “save target as”.**
6. Expecting the file to be downloaded into client’s selected directory.
7. Close the browser window.
8. Repeat the step 1 to 4 using a second client and login to TWiki as TestUser2.
9. Download a file: **right click on the file “Functional_Test_Download_File2.txt”, select “save target as”.**
10. Expecting the file to be downloaded into client’s selected directory.
11. Close the browser window.
12. Annotate completion by initialing box to the left.

2.1.4. Edit Wiki Page Test Instruction (A4)

1. In a new browser window connect to TWiki on Linux server. Type: “**http://192.168.0.200/twiki/bin/view/Main/WebHome”** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: “A4FunctionalTestEdit1”.
5. Edit the A4FunctionalTestEdit1 page: **click “Edit”.**.
6. Modify the wiki page and save the modified wiki page.
7. Expecting successful modification and display of the modified content.
8. Modify the wiki page to its original state and save.
9. Close the browser window.
10. Repeat steps 1to 3 using a second client and login to TWiki as TestUser2.
11. Click on wiki page: “A4FunctionalTestEdit2”.
12. Edit the A4FunctionalTestEdit2 page: **click “Edit”.**.
13. Modify the wiki page and save the modified wiki page.
14. Expecting successful modification and display of the modified content.
15. Modify the wiki page to its original state and save.
16. Close the browser window.
17. Annotate completion by initialing box to the left.

2.1.5. Create new Wiki Page Test Instruction (A5)

1. In a new browser window connect to TWiki on Linux server. Type: “**http://192.168.0.200/twiki/bin/view/Main/WebHome”** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: “A5FunctionalTestCreate”.
5. Edit the A5FunctionalTestCreate page: **click “Edit”.**.
6. Create a link to a new page: type “[[new page1]]” in the edit box.

7. Save the modified wiki page.
8. Click on the "?" link beside the "new page1" text on the displayed content.
9. Type in some text and save the wiki page.
10. Expecting successful modification and display of the modified content.
11. Close the browser window.
12. Repeat the steps 1 to 9 using a second client and login to TWiki as TestUser2, and creating the new page as [[new page2]].
13. Expecting successful modification and display of the modified content.
14. Close the browser window.
15. Annotate completion by initialing box to the left.

2.1.6. Delete Authorized file Test Instruction (A6)

1. In a new browser window connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory.
4. Click on "index", and then click the "NewPage1" topic created in the Create test case.
5. Click on "More topic actions" on the lower right corner.
6. Click "**Delete topic...**, looking for references in *all public webs* **(recommended)**" option.
7. Click "Delete" in the next screen.
8. Expecting successful deletion and display of "new page1?" showing the page does not exist.
9. Close the browser window.
10. Repeat steps 1 to 7 using a second client and login to TWiki as TestUser2, and deleting the "new page2" page.
11. Expecting successful deletion and display of "new page2?" showing the page does not exist.
12. Close the browser window.
13. Annotate completion by initialing box to the left.

2.1.7. Upload Authorized File(s) Test Instruction (A7)

1. In a new browser window connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: "A7FunctionalTestUpload".
5. Create 3 files on client's current directory with distinct filenames.
6. Upload first file to A7FunctionalTestUpload page: click on "Attach", and select the first file using "browse", then click "upload file".

7. Expecting first file uploaded.
8. Upload second and third files.
9. Expecting both files uploaded.
10. Close the browser window.
11. Repeat steps 1 to 8 using a second client and login to TWiki as TestUser2.
12. Expecting all three files uploaded.
13. Close the browser window.
14. On Linux Server, login as root and verify /home/http/htdocs/twiki/pub/Test_Functional/A7FunctionalTestUpload/ directory contains those six files.
15. Close the browser window.
16. Annotate completion by initialing box to the left.

2.1.8. Simultaneous Edit File Test Instruction (A8)

1. In a new browser window connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: "A8FunctionalTestSimultaneousEdit".
5. Edit the wiki page: click **"Edit**", and type in "edited by TestUser1".
6. In new browser window on another client machine connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
7. Login to TWiki as TestUser2.
8. Browse to Test_Functional directory.
9. Edit the same "A8FunctionalTestSimultaneousEdit" wiki page.
10. Expecting a warning message saying TestUser1 is editing the file.
11. Click "Edit anyway" to ignore the warning and continue to edit the file.
12. Edit the wiki page: type in "edited by TestUser2".
13. On the TestUser1 client, click save.
14. Expecting the content to be save successfully.
15. On the TestUser2 client, click save.
16. Expecting a warning message on TestUser2 client saying topic was merged.
17. Click "OK" on TestUser2 client.
18. Expecting the display of the difference of the 2 separate edits.
19. Close the two browser windows.
20. Annotate completion by initialing box to the left.

2.1.9. Setting of Permission on File(s) Test Instruction (A9)

1. In a new browser window connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.

2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: “A9FunctionalTestSetPermission”.
5. Edit the wiki page: click **“Edit”**.
6. Restrict viewing of the page by TestUser1: type “ * Set DENYTOPICVIEW = %MAINWEB%.TestUser2” (without quote, must have at least six space before *).
7. Save the changes.
8. Browse to WebHome of Main page.
9. Logout of TestUser1 and login as TestUser2.
10. Click on wiki page: “A9FunctionalTestSetPermission”.
11. Expecting that access to be denied.
12. Logout of TestUser2 and login as TestUser1, and restore the original permission.
13. Close the browser window.
14. Annotate completion by initialing box to the left.

2.1.10. Changing TWiki Configuration Test Instruction (A10)

1. In a new browser window connect to TWiki on Linux server. Type: “**http://192.168.0.200/twiki/bin/configure”** as the URL.
2. Change the following setting: Statistics->{Stats}{TopViews} to 20 (default = 10)”.
3. Click “Next”.
4. Type in “password” and click **“Save”.**
5. Expecting a “Updating Configuration: setting 1 configuration item” message.
6. Click “return to configuration”.
7. Change the setting back to the original value and save the configuration again.
8. Close the browser window.
9. Annotate completion by initialing box to the left.

2.1.11. Register New User Test Instruction (A11)

1. In a new browser window connect to TWiki on Linux server. Type: “**http://192.168.0.200/twiki/bin/view/Main/WebHome”** as the URL.
2. Click “Register” on sidebar.
3. In the Registration page, key in the following:
    a. First Name: Test
    b. Last Name: User3
    c. Email Address: testuser3@mlsserver.cisrlabmlstestbed1.com
    d. Your Password: password
    e. Retype password: password
    f. Country: USA
4. Click “submit”.

5. Expecting successful registration with a “Thank you for registering” message.
6. Close the browser window.
7. Annotate completion by initialing box to the left.

2.2. Functional Tests (Single Level XTS with NIC configured at sl1:il3)

These tests verify that all functionalities provided by adding TWiki into single level XTS server are behaving correctly.

2.2.1. List Wiki Web/Pages Test Instruction (A1)

1. In a new browser window, connect to TWiki on single level XTS server. Type: “**http://192.168.0.220/twiki/bin/view/Main/WebHome”** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: “A1FunctionalTestList”.
5. Expecting a listing of Functional Test wiki pages A1 to A11.
6. Close the browser window.
7. Repeat the above steps using a second client and login to TWiki as TestUser2.
8. Expecting a listing of Functional Test wiki pages A1 to A11.
9. Close the browser window.
10. Annotate completion by initialing box to the left.

2.2.2. Display Wiki Pages Test Instruction (A2)

1. In a new browser window, connect to TWiki on single level XTS server. Type: “**http://192.168.0.220/twiki/bin/view/Main/WebHome”** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: “A2FunctionalTestDisplay1”.
5. Expecting a display of the content of the wiki page: “This is Display Test 1”.
6. Close the browser window.
7. Repeat the steps 1 to 3 using a second client and login to TWiki as TestUser2.
8. Click on wiki page: “A2FunctionalTestDisplay2”.
9. Expecting a display of the content of the wiki page: “This is Display Test 2”.
10. Close the browser window.
11. Annotate completion by initialing box to the left.

2.2.3. Download File Test Instruction (A3)

1. In a new browser window, connect to TWiki on single level XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar
4. Click on wiki page: "A3FunctionalTestDownload".
5. Download a file: **right click on the file "Functional_Test_Download_File1.txt", select "save target as".**
6. Expecting the file to be downloaded into client's selected directory.
7. Close the browser window.
8. Repeat the step 1 to 4 using a second client and login to TWiki as TestUser2.
9. Download a file: **right click on the file "Functional_Test_Download_File2.txt", select "save target as".**
10. Expecting the file to be downloaded into client's selected directory.
11. Close the browser window.
12. Annotate completion by initialing box to the left.

2.2.4. Edit Wiki Page Test Instruction (A4)

1. In a new browser window connect to TWiki on single level XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: "A4FunctionalTestEdit1".
5. Edit the A4FunctionalTestEdit1 page: click **"Edit".**.
6. Modify the wiki page and save the modified wiki page.
7. Expecting successful modification and display of the modified content.
8. Modify the wiki page to its original state and save.
9. Close the browser window.
10. Repeat steps 1to 3 using a second client and login to TWiki as TestUser2.
11. Click on wiki page: "A4FunctionalTestEdit2".
12. Edit the A4FunctionalTestEdit2 page: click **"Edit".**.
13. Modify the wiki page and save the modified wiki page.
14. Expecting successful modification and display of the modified content.
15. Modify the wiki page to its original state and save.
16. Close the browser window.
17. Annotate completion by initialing box to the left.

2.2.5. Create new Wiki Page Test Instruction (A5)

1. In a new browser window connect to TWiki on single level XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.

2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: “A5FunctionalTestCreate”.
5. Edit the A5FunctionalTestCreate page: click **“Edit”.**.
6. Create a link to a new page: type “[[new page1]]” in the edit box.
7. Save the modified wiki page.
8. Click on the “?” link beside the “new page1” text on the displayed content.
9. Type in some text and save the wiki page.
10. Expecting successful modification and display of the modified content.
11. Close the browser window.
12. Repeat the steps 1 to 9 using a second client and login to TWiki as TestUser2, and creating the new page as [[new page2]].
13. Expecting successful modification and display of the modified content.
14. Close the browser window.
15. Annotate completion by initialing box to the left.

2.2.6. Delete Authorized file Test Instruction (A6)

1. In a new browser window connect to TWiki on single level XTS server. Type: “**http://192.168.0.220/twiki/bin/view/Main/WebHome”** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory.
4. Click on “index”, and then click the “NewPage1” topic created in the Create test case.
5. Click on “More topic actions” on the lower right corner.
6. Click “**Delete topic...**, looking for references in *all public webs* **(recommended)**” option.
7. Click “Delete” in the next screen.
8. Expecting successful deletion and display of “new page1?” showing the page does not exist.
9. Close the browser window.
10. Repeat steps 1 to 7 using a second client and login to TWiki as TestUser2, and deleting the “new page2” page.
11. Expecting successful deletion and display of “new page2?” showing the page does not exist.
12. Close the browser window.
13. Annotate completion by initialing box to the left.

2.2.7. Upload Authorized File(s) Test Instruction (A7)

1. In a new browser window connect to TWiki on single level XTS server. Type: “**http://192.168.0.220/twiki/bin/view/Main/WebHome”** as the URL.
2. Login to TWiki as TestUser1.

3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: “A7FunctionalTestUpload”.
5. Create 3 files on client’s current directory with distinct filenames.
6. Upload first file to A7FunctionalTestUpload page: click on “Attach”, and select the first file using “browse”, then click “upload file”.
7. Expecting first file uploaded.
8. Upload second and third files.
9. Expecting both files uploaded.
10. Close the browser window.
11. Repeat steps 1 to 8 using a second client and login to TWiki as TestUser2.
12. Expecting all three files uploaded.
13. Close the browser window.
14. On Single level XTS Server, login as root and verify /home/http/htdocs/twiki/pub/Test_Functional/A7FunctionalTestUpload/ directory contains those six files.

15. Annotate completion by initialing box to the left.

2.2.8. Simultaneous Edit File Test Instruction (A8)

1. In a new browser window connect to TWiki on single level XTS server. Type: “**http://192.168.0.220/twiki/bin/view/Main/WebHome”** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: “A8FunctionalTestSimultaneousEdit”.
5. Edit the wiki page: click “**Edit**”, and type in “edited by TestUser1”.
6. In new browser window on another client machine connect to TWiki on Single level XTS server. Type: “**http://192.168.0.220/twiki/bin/view/Main/WebHome”** as the URL.
7. Login to TWiki as TestUser2.
8. Browse to Test_Functional directory.
9. Edit the same “A8FunctionalTestSimultaneousEdit” wiki page.
10. Expecting a warning message saying TestUser1 is editing the file.
11. Click “Edit anyway” to ignore the warning and continue to edit the file.
12. Edit the wiki page: type in “edited by TestUser2”.
13. On the TestUser1 client, click save.
14. Expecting the content to be save successfully.
15. On the TestUser2 client, click save.
16. Expecting a warning message on TestUser2 client saying topic was merged.
17. Click “OK” on TestUser2 client.
18. Expecting the display of the difference of the 2 separate edits.
19. Close the two browser windows.

20. Annotate completion by initialing box to the left.

2.2.9. Setting of Permission on File(s) Test Instruction (A9)

1. In a new browser window connect to TWiki on single level XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Browse to Test_Functional directory using sidebar.
4. Click on wiki page: "A9FunctionalTestSetPermission".
5. Edit the wiki page: click "**Edit**".
6. Restrict viewing of the page by TestUser1: type " * Set DENYTOPICVIEW = %MAINWEB%.TestUser2" (without quote, must have at least six space before *).
7. Save the changes.
8. Browse to WebHome of Main page.
9. Logout of TestUser1 and login as TestUser2.
10. Click on wiki page: "A9FunctionalTestSetPermission".
11. Expecting that access to be denied.
12. Logout of TestUser2 and login as TestUser1, and restore the original permission.
13. Close the browser window.
14. Annotate completion by initialing box to the left.

2.2.10. Changing TWiki Configuration Test Instruction (A10)

1. In a new browser window connect to TWiki on single level XTS server. Type: "**http://192.168.0.220/twiki/bin/configure"** as the URL.
2. Change the following setting: Statistics->{Stats}{TopViews} to 20 (default = 10)".
3. Click "Next".
4. Type in "password" and click "**Save**".
5. Expecting a "Updating Configuration: setting 1 configuration item" message.
6. Click "return to configuration".
7. Change the setting back to the original value and save the configuration again.
8. Close the browser window.
9. Annotate completion by initialing box to the left.

2.2.11. Register New User Test Instruction (A11)

1. In a new browser window connect to TWiki on single level XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.
2. Click "Register" on sidebar.
3. In the Registration page, key in the following:
    g. First Name: Test
    h. Last Name: User3
    i. Email Address: testuser3@mlsserver.cisrlabmlstestbed1.com

j. Your Password: password
k. Retype password: password
l. Country: USA

4. Click “submit”.
5. Expecting successful registration with a “Thank you for registering” message.
6. Close the browser window.
7. Annotate completion by initialing box to the left.

2.3. Functional Tests (Multilevel XTS)

These tests verify that all functionalities provided by adding TWiki into MYSEA Testbed are behaving correctly.

2.3.1. List Wiki Web/Pages Test Instruction (A1)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window, connect to TWiki on multilevel XTS server. Type: “**http://192.168.0.130/twiki/bin/view/Main/WebHome”** as the URL.
3. Login to TWiki as TestUser1.
4. Browse to Test_Functional directory using sidebar.
5. Click on wiki page: “A1FunctionalTestList”.
6. Expecting a listing of Functional Test wiki pages A1 to A11.
7. Close the browser window.
8. Repeat steps 1 to 5 using a second client, login to XTS as mdemo2 and login to TWiki as TestUser2.
9. Expecting a listing of Functional Test wiki pages A1 to A11.
10. Close the browser window.
11. Annotate completion by initialing box to the left.

2.3.2. Display Wiki Pages Test Instruction (A2)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window, connect to TWiki on multilevel XTS server. Type: “**http://192.168.0.130/twiki/bin/view/Main/WebHome”** as the URL.
3. Login to TWiki as TestUser1.
4. Browse to Test_Functional directory using sidebar.
5. Click on wiki page: “A2FunctionalTestDisplay1”.
6. Expecting a display of the content of the wiki page: “This is Display Test 1”.
7. Close the browser window.

8. Repeat steps 1 to 4 using a second client, login to XTS as mdemo2, and login to TWiki as TestUser2.
9. Click on wiki page: "A2FunctionalTestDisplay2".
10. Expecting a display of the content of the wiki page: "This is Display Test 2".
11. Close the browser window.
12. Annotate completion by initialing box to the left.

2.3.3. Download File Test Instruction (A3)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window, connect to TWiki on multilevel XTS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Browse to Test_Functional directory using sidebar
5. Click on wiki page: "A3FunctionalTestDownload".
6. Download a file: **right click on the file "Functional_Test_Download_File1.txt", select "save target as".**
7. Expecting the file to be downloaded into client's selected directory.
8. Close the browser window.
9. Repeat steps 1 to 5 using a second client, login to XTS as mdemo2, and login to TWiki as TestUser2.
10. Download a file: **right click on the file "Functional_Test_Download_File2.txt", select "save target as".**
11. Expecting the file to be downloaded into client's selected directory.
12. Close the browser window.
13. Annotate completion by initialing box to the left.

2.3.4. Edit Wiki Page Test Instruction (A4)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on multilevel XTS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Browse to Test_Functional directory using sidebar.
5. Click on wiki page: "A4FunctionalTestEdit1".
6. Edit the A4FunctionalTestEdit1 page: **click "Edit".**.
7. Modify the wiki page and save the modified wiki page.
8. Expecting successful modification and display of the modified content.
9. Modify the wiki page to its original state and save.
10. Close the browser window.

11. Repeat steps 1to 4 using a second client, login to XTS as mdemo2, and login to TWiki as TestUser2.
12. Click on wiki page: “A4FunctionalTestEdit2”.
13. Edit the A4FunctionalTestEdit2 page: **click “Edit”.**.
14. Modify the wiki page and save the modified wiki page.
15. Expecting successful modification and display of the modified content.
16. Modify the wiki page to its original state and save.
17. Close the browser window.
18. Annotate completion by initialing box to the left.

2.3.5. Create new Wiki Page Test Instruction (A5)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on multilevel XTS server. Type: “**http://192.168.0.130/twiki/bin/view/Main/WebHome”** as the URL.
3. Login to TWiki as TestUser1.
4. Browse to Test_Functional directory using sidebar.
5. Click on wiki page: “A5FunctionalTestCreate”.
6. Edit the A5FunctionalTestCreate page: **click “Edit”.**.
7. Create a link to a new page: type “[[new page1]]” in the edit box.
8. Save the modified wiki page.
9. Click on the “?” link beside the “new page1” text on the displayed content.
10. Type in some text and save the wiki page.
11. Expecting successful modification and display of the modified content.
12. Close the browser window.
13. Repeat the steps 1 to 10 using a second client, login to XTS as mdemo2, and login to TWiki as TestUser2, and creating the new page as [[new page2]].
14. Expecting successful modification and display of the modified content.
15. Close the browser window.
16. Annotate completion by initialing box to the left.

2.3.6. Delete Authorized file Test Instruction (A6)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on multilevel XTS server. Type: “**http://192.168.0.130/twiki/bin/view/Main/WebHome”** as the URL.
3. Login to TWiki as TestUser1.
4. Browse to Test_Functional directory.
5. Click on “index”, and then click the “NewPage1” topic created in the Create test case.
6. Click on “More topic actions” on the lower right corner.

7. Click "**Delete topic...**, looking for references in *all public webs* **(recommended)**" option.
8. Click "Delete" in the next screen.
9. Expecting successful deletion and display of "new page1?" showing the page does not exist.
10. Close the browser window.
11. Repeat steps 1 to 8 using a second client, login to XTS as mdemo2, and login to TWiki as TestUser2, and deleting the "new page2" page.
12. Expecting successful deletion and display of "new page2?" showing the page does not exist.
13. Close the browser window.
14. Annotate completion by initialing box to the left.

2.3.7. Upload Authorized File(s) Test Instruction (A7)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on multilevel XTS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Browse to Test_Functional directory using sidebar.
5. Click on wiki page: "A7FunctionalTestUpload".
6. Create 3 files on client's current directory with distinct filenames.
7. Upload first file to A7FunctionalTestUpload page: click on "Attach", and select the first file using "browse", then click "upload file".
8. Expecting first file uploaded.
9. Upload second and third files.
10. Expecting both files uploaded.
11. Close the browser window
12. Repeat steps 1 to 9 using a second client, login to XTS as mdemo2, and login to TWiki as TestUser2.
13. Expecting all three files uploaded.
14. Close the browser window.
15. On Multilevel XTS Server, login as admin at (sl1:il0) and verify /home/http/htdocs/twiki/pub/SIM_UNCLASSIFIED/Test_Functional/A7FunctionalTestUpload/ directory contains those six files.
16. Annotate completion by initialing box to the left.

2.3.8. Simultaneous Edit File Test Instruction (A8)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.

2. In a new browser window connect to TWiki on multilevel XTS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Browse to Test_Functional directory using sidebar.
5. Click on wiki page: "A8FunctionalTestSimultaneousEdit".
6. Edit the wiki page: click "Edit", and type in "edited by TestUser1".
7. Establish a SIM_UNCLASSIFIED session for mdemo2 user using the TCBE, for another client machine.
8. In new browser window on the new client machine connect to TWiki on Multilevel XTS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
9. Login to TWiki as TestUser2.
10. Browse to Test_Functional directory.
11. Edit the same "A8FunctionalTestSimultaneousEdit" wiki page.
12. Expecting a warning message saying TestUser1 is editing the file.
13. Click "Edit anyway" to ignore the warning and continue to edit the file.
14. Edit the wiki page: type in "edited by TestUser2".
15. On the TestUser1 client, click save.
16. Expecting the content to be save successfully.
17. On the TestUser2 client, click save.
18. Expecting a warning message on TestUser2 client saying topic was merged.
19. Click "OK" on TestUser2 client.
20. Expecting the display of the difference of the 2 separate edits.
21. Close the two browser windows.
22. Annotate completion by initialing box to the left.

2.3.9. Setting of Permission on File(s) Test Instruction (A9)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on multilevel XTS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Browse to Test_Functional directory using sidebar.
5. Click on wiki page: "A9FunctionalTestSetPermission".
6. Edit the wiki page: click "**Edit**".
7. Restrict viewing of the page by TestUser1: type " * Set DENYTOPICVIEW = %MAINWEB%.TestUser2"(without quote, must have at least six space before *).
8. Save the changes.
9. Browse to WebHome of Main page.
10. Logout of TestUser1 from TWiki and logout of mdemo1 from XTS.
11. Login to XTS as mdemo2 using the TCBE and login to TWiki as TestUser2.
12. Browse to Test_Functional directory using sidebar.
13. Click on wiki page: "A9FunctionalTestSetPermission".

14. Expecting that access to be denied.
15. Logout of TestUser2 and login as TestUser1, and restore the original permission.
16. Close the browser window.
17. Annotate completion by initialing box to the left.

2.3.10. Changing TWiki Configuration Test Instruction (A10)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on multilevel XTS server. Type: "**http://192.168.0.130/twiki/bin/configure"** as the URL.
3. Change the following setting: Statistics->{Stats}{TopViews} to 20 (default = 10)".
4. Click "Next".
5. Type in "password" and click "**Save**".
6. Expecting a "Updating Configuration: setting 1 configuration item" message.
7. Click "return to configuration".
8. Change the setting back to the original value and save the configuration again.
9. Close the browser window.
10. Annotate completion by initialing box to the left.

2.3.11. Register New User Test Instruction (A11)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on multilevel XTS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Click "Register" on sidebar.
4. In the Registration page, key in the following:
    m. First Name: Test
    n. Last Name: User10
    o. Email Address: testuser10@mlsserver.cisrlabmlstestbed1.com
    p. Your Password: password
    q. Retype password: password
    r. Country: USA
5. Click "submit".
6. Expecting successful registration with a "Thank you for registering" message.
7. Close the browser window.
8. Annotate completion by initialing box to the left.

2.4. DAC Tests (Linux)

2.4.1. TWiki DenyTopicView Permission Test Instruction (BL1)

1. In a new browser window connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Access the BL1 test page on TestView directory by typing: "**http://192.168.0.200/twiki/bin/view/Test_DAC_Linux/BL1/TestTopic**" as the URL.
4. Expecting access to be denied, with an error message saying "Access Denied".
5. Close the browser window.
6. Repeat steps 1 to 3, logging in as TestUser2.
7. Expecting access to be denied, with an error message saying "Access Denied".
8. Close the browser window.
9. Repeat steps 1 to 3, logging in as AdminUser.
10. Expecting access to be granted, displaying content: "Test Text".
11. Close the browser window.
12. Annotate completion by initialing box to the left.

2.4.2. TWiki AllowTopicView Permission Test Instruction (BL2)

1. In a new browser window connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Access the BL2 test page on TestView directory by typing: "**http://192.168.0.200/twiki/bin/view/Test_DAC_Linux/BL2/TestTopic**" as the URL.
4. Expecting access to be granted for TestTopic with content "Test Text", and with the following message on the leftbar: "No permission to view Test_DAC_Linux/BL2.WebLeftBar" .
5. Close the browser window.
6. Repeat steps 1 to 3, logging in as TestUser2.
7. Expecting access to be denied, with an error message saying "Access Denied".
8. Close the browser window.
9. Repeat steps 1 to 3, logging in as AdminUser.
10. Expecting access to be granted with content: "Test Text".
11. Close the browser window.
12. Annotate completion by initialing box to the left.

2.4.3. TWiki AllowWebChange Permission Test Instruction (BL3)

1. In a new browser window connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Access the BL3 test page on TestChange directory by typing: "**http://192.168.0.200/twiki/bin/view/Test_DAC_Linux/BL3/TestTopic**" as the URL.
4. Click "Edit", insert some text, and click "**Save**".
5. Expecting changes be saved successfully.
6. Close the browser window.
7. Repeat steps 1 to 4, logging in as TestUser2.
8. Expecting change access to be denied, with an error message saying "Access Denied".
9. Close the browser window.
10. Repeat steps 1 to 4, logging in as AdminUser.
11. Insert some text, and click "**Save**".
12. Expecting changes be saved successfully.
13. Close the browser window.
14. Annotate completion by initialing box to the left.

2.4.4. TWiki DenyWebRename Permission Test Instruction (BL4)

1. In a new browser window connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Access the BL4 test page on TestChange directory by typing: "**http://192.168.0.200/twiki/bin/view/Test_DAC_Linux/BL4/TestTopic**" as the URL.
4. Attempt to rename the TestTopic file: click "More topic actions", then click "Rename/move topic… looking for references in *all public webs* **(recommended)**".
5. Expecting the rename operation to be denied, with an "Access Denied" message.
6. Close the browser window.
7. Repeat steps 1 to 4, logging in as TestUser2.
8. Expecting the rename operation to be denied, with an error message saying "Access Denied".
9. Close the browser window.
10. Repeat steps 1 to 4, logging in as AdminUser.
11. In the "To topic" field, enter "TestTopic1" as the new topic name.
12. Click "Rename/move" at the bottom of the page.
13. Expecting the rename operation to be successful.
14. Close the browser window.
15. Annotate completion by initialing box to the left.

2.4.5. TWiki View DAC Not Overriding OS DAC Test Instruction (BL5)

1. In a new browser window connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Access the BL5 test page on TestView directory by typing: "**http://192.168.0.200/twiki/bin/view/Test_DAC_Linux/BL5/TestTopic**" as the URL.
4. Expecting access to be granted, but content of TestTopic, i.e.. "Test text", is not displayed.
5. Close the browser window.
6. Repeat steps 1 to 3, logging in as TestUser2.
7. Expecting access to be granted, but content of TestTopic, i.e.. "Test text", is not displayed.
8. Close the browser window.
9. Repeat steps 1 to 3, logging in as AdminUser.
10. Expecting access to be granted, but content of TestTopic, i.e.. "Test text", is not displayed.
11. Close the browser window.
12. Annotate completion by initialing box to the left.

2.4.6. TWiki Change DAC Not Overriding OS DAC Test Instruction (BL6)

1. In a new browser window connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Access the BL6 test page on TestChange directory by typing: "**http://192.168.0.200/twiki/bin/view/Test_DAC_Linux/BL6/TestTopic**" as the URL.
4. Click "**Edit**", type in some text, and click "**Save**".
5. Expecting operation to be denied, with an error message saying: "RCS: failed to create file path: Permission denied".
6. Close the browser window.
7. Repeat steps 1 to 4, logging in as TestUser2.
8. Expecting operation to be denied, with an error message saying: "RCS: failed to create file path: Permission denied".
9. Close the browser window.
10. Repeat steps 1 to 4, logging in as AdminUser.
11. Expecting operation to be denied, with an error message saying: "RCS: failed to create file path: Permission denied".
12. Close the browser window.
13. Annotate completion by initialing box to the left.

2.4.7. TWiki Rename DAC Not Overriding OS DAC Test Instruction (BL7)

1. In a new browser window connect to TWiki on Linux server. Type: "**http://192.168.0.200/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Access the BL7 test page on TestRename directory by typing: "**http://192.168.0.200/twiki/bin/view/Test_DAC_Linux/BL7/TestTopic**" as the URL.
4. Attempt to rename the TestTopic file: click "More topic actions", then click "Rename/move topic… looking for references in *all public webs* **(recommended)**".
5. Type "TestTopic1" as the new topic name, and click "Rename/move" at the bottom of the page.
6. Expecting the rename operation to be denied, with an error message saying "During renaming of topic, an error was found. Please notify your TWiki administrator".
7. Close the browser window.
8. Repeat steps 1 to 5, logging in as TestUser2.
9. Expecting the rename operation to be denied, with an error message saying "During renaming of topic, an error was found. Please notify your TWiki administrator".
10. Close the browser window.
11. Repeat steps 1 to 5, logging in as AdminUser.
12. Expecting the rename operation to be denied, with an error message saying "During renaming of topic, an error was found. Please notify your TWiki administrator".
13. Close the browser window.
14. Annotate completion by initialing box to the left.

2.5. DAC Tests (Single Level XTS with NIC configured at sl1:il3)

2.5.1. TWiki DenyTopicView Permission Test Instruction (BS1) (MAC Read Equal)

1. In a new browser window connect to TWiki on Single level XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Access the BS1 test page on TestView directory by typing: "**http://192.168.0.220/twiki/bin/view/Test_DAC_SingleLevel/BS1/TestTopic**" as the URL.
4. Expecting access to be denied, with an error message saying "Access Denied".
5. Close the browser window.
6. Repeat steps 1 to 3, logging in as TestUser2.
7. Expecting access to be denied, with an error message saying "Access Denied".

8. Close the browser window.
9. Repeat steps 1 to 3, logging in as AdminUser.
10. Expecting access to be granted, displaying content: “Test Text”.
11. Close the browser window.
12. Annotate completion by initialing box to the left.

2.5.2. TWiki AllowTopicView Permission Test Instruction (BS2) (MAC Read Down)

1. In a new browser window connect to TWiki on Single level XTS server. Type: “**http://192.168.0.220/twiki/bin/view/Main/WebHome”** as the URL.
2. Login to TWiki as TestUser1.
3. Access the BS2 test page on TestView directory by typing: “**http://192.168.0.220/twiki/bin/view/Test_DAC_SingleLevel/BS2/TestTopic**” as the URL.
4. Expecting access to be granted for TestTopic with content “Test Text”, and with the following message on the leftbar: “No permission to view Test_DAC_SingleLevel/BS2.WebLeftBar” .
5. Close the browser window.
6. Repeat steps 1 to 3, logging in as TestUser2.
7. Expecting access to be denied, with an error message saying “Access Denied”.
8. Close the browser window.
9. Repeat steps 1 to 3, logging in as AdminUser.
10. Expecting access to be granted, with content: “Test Text”.
11. Close the browser window.
12. Annotate completion by initialing box to the left.

2.5.3. TWiki AllowWebChange Permission Test Instruction (BS3) (MAC Write Equal)

1. In a new browser window connect to TWiki on Single level XTS server. Type: “**http://192.168.0.220/twiki/bin/view/Main/WebHome”** as the URL.
2. Login to TWiki as TestUser1.
3. Access the BS3 test page on TestChange directory by typing: “**http://192.168.0.220/twiki/bin/view/Test_DAC_SingleLevel/BS3/TestTopic**” as the URL.
4. Click “**Edit**”, insert some text, and click “**Save**”.
5. Expecting changes be saved successfully.
6. Close the browser window.
7. Repeat steps 1 to 4, logging in as TestUser2.
8. Expecting change access to be denied, with an error message saying “Access Denied”.
9. Close the browser window.

10. Repeat steps 1 to 4, logging in as AdminUser.
11. Insert some text, and click "**Save**".
12. Expecting changes be saved successfully.
13. Close the browser window.
14. Annotate completion by initialing box to the left.

2.5.4. TWiki DenyWebRename Permission Test Instruction (BS4) (Write Equal)

1. In a new browser window connect to TWiki on Multilevel XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Access the BS4 test page on TestChange directory by typing: "**http://192.168.0.220/twiki/bin/view/Test_DAC_SingleLevel/BS4/TestTopic**" as the URL.
4. Attempt to rename the TestTopic file: click "More topic actions", then click "Rename/move topic… looking for references in *all public webs* **(recommended)**".
5. Expecting the rename operation to be denied, with an "Access Denied" message.
6. Close the browser window.
7. Repeat steps 1 to 4, logging in as TestUser2.
8. Expecting the rename operation to be denied, with an "Access Denied" message.
9. Close the browser window.
10. Repeat steps 1 to 4, logging in as AdminUser.
11. In the "To topic" field, enter "TestTopic1" as the new topic name.
12. Click "Rename/move" at the bottom of the page.
13. Expecting the rename operation to be successful.
14. Close the browser window.
15. Annotate completion by initialing box to the left.

2.5.5. TWiki View DAC Not Overriding OS DAC Test Instruction (BS5) (MAC Read Equal)

1. In a new browser window connect to TWiki on Single level XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Access the BS5 test page on TestView directory by typing: "**http://192.168.0.220/twiki/bin/view/Test_DAC_SingleLevel/BS5/TestTopic**" as the URL.
4. Expecting access to be granted, but content of TestTopic, i.e.. "Test text", is not displayed.
5. Close the browser window.
6. Repeat steps 1 to 3, logging in as TestUser2.

7. Expecting access to be granted, but content of TestTopic, i.e.. “Test text”, is not displayed.
8. Close the browser window.
9. Repeat steps 1 to 3, logging in as AdminUser.
10. Expecting access to be granted, but content of TestTopic, i.e.. “Test text”, is not displayed.
11. Close the browser window.
12. Annotate completion by initialing box to the left.

2.5.6. TWiki Change DAC Not Overriding OS DAC Test Instruction (BS6) (MAC Write Equal)

1. In a new browser window connect to TWiki on Single level XTS server. Type: “**http://192.168.0.220/twiki/bin/view/Main/WebHome”** as the URL.
2. Login to TWiki as TestUser1.
3. Access the BS6 test page on TestChange directory by typing: “**http://192.168.0.220/twiki/bin/view/Test_DAC_SingleLevel/BS6/TestTopic**” as the URL.
4. Click “**Edit**”, type in some text, and click “**Save**”.
5. Expecting operation to be denied, with an error message saying: “RCS: failed to create file path: Permission denied”.
6. Close the browser window.
7. Repeat steps 1 to 4, logging in as TestUser2.
8. Expecting operation to be denied, with an error message saying: “RCS: failed to create file path: Permission denied”.
9. Close the browser window.
10. Repeat steps 1 to 4, logging in as AdminUser.
11. Expecting operation to be denied, with an error message saying: “RCS: failed to create file path: Permission denied”.
12. Close the browser window.
13. Annotate completion by initialing box to the left.

2.6. DAC Tests (Multilevel Level XTS)

2.6.1. TWiki DenyTopicView Permission Test Instruction (BM1) (Read Equal)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on multilevel XTS server. Type: “**http://192.168.0.130/twiki/bin/view/Main/WebHome”** as the URL.
3. Login to TWiki as TestUser1.

4. Access the BM1 test page on TestView directory by typing: "**http://192.168.0.130/twiki/bin/view/SIM_UNCLASSIFIED /Test_DAC_MultiLevel/BM1/TestTopic**" as the URL.
5. Expecting access to be denied, with an error message saying "Access Denied".
6. Close the browser window.
7. Repeat steps 1 to 4, logging in as mdemo2 and TestUser2.
8. Expecting access to be denied, with an error message saying "Access Denied".
9. Close the browser window.
10. Repeat steps 1 to 4, logging in as mdemo2 and AdminUser.
11. Expecting access to be granted, displaying content: "Test Text".
12. Close the browser window.
13. Annotate completion by initialing box to the left.

2.6.2. TWiki AllowTopicView Permission Test Instruction (BM2) (Read Down)

1. Establish a SIM_SECRET session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on multilevel XTS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Access the BM2 test page on TestView directory by typing: "**http://192.168.0.130/twiki/bin/view/SIM_UNCLASSIFIED /Test_DAC_MultiLevel/BM2/TestTopic**" as the URL.
5. Expecting access to be granted for TestTopic with content "Test Text", and with the following message on the leftbar: "No permission to view Test_DAC_SingleLevel/BS2.WebLeftBar" .
6. Close the browser window.
7. Repeat steps 1 to 4, logging in as mdemo2 and TestUser2.
8. Expecting access to be denied, with an error message saying "Access Denied".
9. Close the browser window.
10. Repeat steps 1 to 4, logging in as mdemo2 and AdminUser.
11. Expecting access to be granted, with content :"Test Text".
12. Close the browser window.
13. Annotate completion by initialing box to the left.

2.6.3. TWiki AllowWebChange Permission Test Instruction (BM3) (MAC Write Equal)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on multilevel XTS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.

3. Login to TWiki as TestUser1.
4. Access the BM3 test page on TestChange directory by typing: "**http://192.168.0.130/twiki/bin/view/SIM_UNCLASSIFIED/Test_DAC_MultiLevel/BM3/TestTopic**" as the URL.
5. Click "Edit", insert some text, and click "Save".
6. Expecting changes be saved successfully.
7. Close the browser window.
8. Repeat steps 1 to 5, logging in as mdemo2 and TestUser2.
9. Expecting change access to be denied, with an error message saying "Access Denied".
10. Close the browser window.
11. Repeat steps 1 to 5, logging in as mdemo2 and AdminUser.
12. Insert some text, and click "**Save**".
13. Expecting changes be saved successfully.
14. Close the browser window.
15. Annotate completion by initialing box to the left.

2.6.4. TWiki DenyWebRename Permission Test Instruction (BM4) (Write Equal)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on Multilevel XTS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Access the BM4 test page on TestChange directory by typing: "**http://192.168.0.130/twiki/bin/view/SIM_UNCLASSIFIED /Test_DAC_MultiLevel/BM4/TestTopic**" as the URL.
5. Attempt to rename the TestTopic file: click "More topic actions", then click "Rename/move topic… looking for references in *all public webs* **(recommended)**".
6. Expecting the rename operation to be denied, with an "Access Denied" message.
7. Close the browser window.
8. Repeat steps 1 to 5, logging in as mdemo2 and TestUser2.
9. Expecting the rename operation to be denied, with an "Access Denied" message.
10. Close the browser window.
11. Repeat steps 1 to 5, logging in as mdemo2 and AdminUser.
12. In the "To topic" field, enter "TestTopic1" as the new topic name.
13. Click "Rename/move" at the bottom of the page.
14. Expecting the rename operation to be successful.
15. Close the browser window.
16. Annotate completion by initialing box to the left.

2.6.5. TWiki View DAC Not Overriding OS DAC Test Instruction (BM5) (Read Down)

1. Establish a SIM_SECRET session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on Multilevel XTS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Access the BM5 test page on TestView directory by typing: "**http://192.168.0.130/twiki/bin/view/SIM_UNCLASSIFIED /Test_DAC_MultiLevel/BM5/TestTopic**" as the URL.
5. Expecting access to be granted, but content of TestTopic, i.e.. "Test text", is not displayed.
6. Close the browser window.
7. Repeat steps 1 to 4, logging in as mdemo2 and TestUser2.
8. Expecting access to be granted, but content of TestTopic, i.e.. "Test text", is not displayed.
9. Close the browser window.
10. Repeat steps 1 to 4, logging in as mdemo2 and AdminUser.
11. Expecting access to be granted, but content of TestTopic, i.e.. "Test text", is not displayed.
12. Close the browser window.
13. Annotate completion by initialing box to the left.

2.6.6. TWiki Rename DAC Not Overriding OS DAC Test Instruction (BM6) (Write equal)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on Multilevel XTS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Access the BM6 test page on TestRename directory by typing: "**http://192.168.0.130/twiki/bin/view/SIM_UNCLASSIFIED /Test_DAC_MultiLevel/BM6/TestTopic**" as the URL.
5. Attempt to rename the TestTopic file: click "More topic actions", then click "Rename/move topic… looking for references in *all public webs* **(recommended)**".
6. Type "TestTopic1" as the new topic name, and click "Rename/move" at the bottom of the page.
7. Expecting the rename operation to be denied, with an error message saying "During renaming of topic, an error was found. Please notify your TWiki administrator".
8. Close the browser window.

9. Repeat steps 1 to 6, logging in as mdemo2 and TestUser2.
10. Expecting the rename operation to be denied, with an error message saying "During renaming of topic, an error was found. Please notify your TWiki administrator".
11. Close the browser window.
12. Repeat steps 1 to 6, logging in as mdemo2 and AdminUser.
13. Expecting the rename operation to be denied, with an error message saying "During renaming of topic, an error was found. Please notify your TWiki administrator".
14. Close the browser window.
15. Annotate completion by initialing box to the left.

2.7. MAC Test (Single Level XTS with NIC configured at sl1:il3)

2.7.1. Display Unauthorized Wiki Pages (MAC read up & DAC allowed) Test Instruction (CS1)

1. In a new browser window connect to TWiki on single level XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. View a sl2:il3 page. Type: "**http://192.168.0.220/twiki/bin/view/Test_MAC_SingleLevel/CS1/TestTopic"** as the URL.
4. Expecting an error message saying "The Test_MAC_SingleLevel/CS1 web does not exist".
5. Close the browser window.
6. Annotate completion by initialing box to the left.

2.7.2. Display Authorized Wiki Pages (MAC read down & DAC allowed) Test Instruction (CS2)

1. In a new browser window connect to TWiki on single level XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. View a sl0:il3 page. Type: "**http://192.168.0.220/twiki/bin/view/Test_MAC_SingleLevel/CS2/TestTopic"** as the URL.
4. Expecting a display of the wiki page, with content: "Test Text".
5. Close the browser window.
6. Annotate completion by initialing box to the left.

2.7.3. Display Authorized Wiki Pages (MAC read equal & DAC allowed) Test Instruction (CS3)

1. In a new browser window connect to TWiki on single level XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. View a sl1:il3 page. Type: "**http://192.168.0.220/twiki/bin/view/Test_MAC_SingleLevel/CS3/TestTopic"** as the URL.
4. Expecting a display of the wiki page, with content: "Test Text".
5. Close the browser window.
6. Annotate completion by initialing box to the left.

2.7.4. Edit Unauthorized File (MAC write up & DAC allowed) Test Instruction (CS4)

1. In a new browser window connect to TWiki on single level XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Attempt to edit a sl2:il3 page. Type: "**http://192.168.0.220/twiki/bin/view/Test_MAC_SingleLevel/CS4/TestTopic"** as the URL.
4. Expecting an error message saying "The Test_MAC_SingleLevel/CS4 web does not exist".
5. Close the browser window.
6. Annotate completion by initialing box to the left.

2.7.5. Edit Unauthorized File (MAC write down & DAC allowed) Test Instruction (CS5)

1. In a new browser window connect to TWiki on single level XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Attempt to edit a sl0:il3 page. Type: "**http://192.168.0.220/twiki/bin/view/Test_MAC_SingleLevel/CS5/TestTopic"** as the URL, then click **"Edit"**.
4. Expecting error message saying "RCS: failed to create file path: permission denied".
5. Close the browser window.
6. Annotate completion by initialing box to the left.

2.7.6. Edit Authorized Wiki Page (MAC write equal & DAC allowed) Test Instruction (CS6)

1. In a new browser window connect to TWiki on single level XTS server. Type: "**http://192.168.0.220/twiki/bin/view/Main/WebHome"** as the URL.
2. Login to TWiki as TestUser1.
3. Attempt to edit a sl1:il3 page. Type: "**http://192.168.0.220/twiki/bin/view/Test_MAC_SingleLevel/CS6/TestTopic"** as the URL, then click **"Edit"**.
4. Modify the wiki page and save the modified wiki page.
5. Expecting successful modification and display of the modified content.
6. Modify the wiki page to its original state and save.
7. Close the browser window.
8. Annotate completion by initialing box to the left.

2.8. MAC Test (Multilevel XTS)

2.8.1. Display Unauthorized Wiki Pages (MAC read up & DAC allowed) Test Instruction (CM1)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using TCBE, for the client machine.
2. In a new browser window connect to TWiki on MLS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. View a SIM_SECRET page. Type: "**http://192.168.0.130/twiki/bin/view/SIM_SECRET/Test_MAC_MultiLevel/CM1/TestTopic"** as the URL.
5. Expecting an error message saying "The SIM_SECRET/Test_MAC_MultiLevel/CM1 web does not exist".
6. Close the browser window.
7. Annotate completion by initialing box to the left.

2.8.2. Display Authorized Wiki Pages (MAC read down & DAC allowed) Test Instruction (CM2)

1. Establish a SIM_SECRET session for mdemo1 user using TCBE, for the client machine.
2. In a new browser window connect to TWiki on MLS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. View a SIM_UNCLASSIFIED page. Type:

"**http://192.168.0.130/twiki/bin/view/SIM_UNCLASSIFIED/Test_MAC_MultiLevel/CM2/TestTopic"** as the URL.

5. Expecting a display of the wiki page, with content: "Test Text".
6. Close the browser window.
7. Annotate completion by initialing box to the left.

2.8.3. Display Authorized Wiki Pages (MAC read equal & DAC allowed) Test Instruction (CM3)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using TCBE, for the client machine.
2. In a new browser window connect to TWiki on MLS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. View a SIM_UNCLASSIFIED page. Type: "**http://192.168.0.130/twiki/bin/view/SIM_UNCLASSIFIED/Test_MAC_MultiLevel/CM3/TestTopic"** as the URL.
5. Expecting a display of the wiki page, with content: "Test Text".
6. Close the browser window.
7. Annotate completion by initialing box to the left.

2.8.4. Edit Unauthorized File (MAC write up & DAC allowed) Test Instruction (CM4)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on MLS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Attempt to edit a SIM_SECRET page. Type: "**http://192.168.0.130/twiki/bin/view/SIM_SECRET/Test_MAC_MultiLevel/CM4/TestTopic"** as the URL.
5. Expecting an error message saying "The SIM_SECRET/Test_MAC_MultiLevel/CM4 web does not exist".
6. Close the browser window.
7. Annotate completion by initialing box to the left.

2.8.5. Edit Unauthorized File (MAC write down & DAC allowed) Test Instruction (CM5)

1. Establish a SIM_SECRET session for mdemo1 user using the TCBE, for the client machine.

2. In a new browser window connect to TWiki on MLS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Attempt to edit a SIM_UNCLASSIFIED page. Type: "**http://192.168.0.130/twiki/bin/view/SIM_UNCLASSIFIED/Test_MAC_MultiLevel/CM5/TestTopic"** as the URL.
5. Click **"Edit"**.
6. Expecting error message saying: "RCS: failed to create file path: Permission denied".
7. Close the browser window.
8. Annotate completion by initialing box to the left.

2.8.6. Edit Authorized Wiki Page (MAC write equal & DAC allowed) Test Instruction (CM6)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on MLS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Attempt to edit a SIM_UNCLASSIFIED page. Type: "**http://192.168.0.130/twiki/bin/view/SIM_UNCLASSIFIED/Test_MAC_MultiLevel/CM6/TestTopic"** as the URL.
5. Click **"Edit"**, type some text, and click **"Save"**.
6. Expecting successful modification and display of the modified content.
7. Modify the wiki page to its original state and save.
8. Close the browser window.
9. Annotate completion by initialing box to the left.

2.8.7. Link Unauthorized Wiki Page (MAC write equal, read up & DAC allowed) Test Instruction (CM7)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user using the TCBE, for the client machine.
2. In a new browser window connect to TWiki on MLS server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Attempt to edit a SIM_UNCLASSIFIED page. Type: "**http://192.168.0.130/twiki/bin/view/SIM_UNCLASSIFIED/Test_MAC_MultiLevel/CM7/TestTopic"** as the URL.
5. Click **"Edit"**, type the following (without quote): "[[SIM_SECRET.CM4.TestTopic][secretlink]]", and click **"Save"**.
6. Expecting successful modification and display of the modified content.

7. Click on the “secretlink?’’ link.
8. Expecting a message saying “The SIM_SECRET/CM4 web does not exist”. Click “Create a new web” at the bottom of the page.
9. In the “Adding a New Web” page, click “create new web”.
10. Expecting an error message saying Access Denied.
11. Repeat steps 1 to 9, logging in as mdemo2 and AdminUser.
12. Expecting an error message saying “RCS: failed to create file path: Permission denied”.
13. Modify the wiki page to its original state and save.
14. Close the browser window.
15. Annotate completion by initialing box to the left.

2.9. Integration Tests

2.9.1. Switch to Wiki Test Instruction (D1)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user to MYSEA MLS server using the TCBE, for the client machine.
2. In a new explorer window connect to WebDAV on the MYSEA MLS server. Type: “**\\192.168.0.131\dav”** as the URL.
3. Verify that it is working by editing any file.
4. Logout from the MYSEA MLS server. Type: “logout” at the TCBE.
5. Close the explorer window.
6. Establish a SIM_UNCLASSIFIED session for mdemo1 user to wiki server using the TCBE, for the Client machine.
7. In a new browser window connect to TWiki on the server. Type: “**http://192.168.0.130/twiki/bin/view/Main/WebHome”** as the URL.
8. Login to TWiki as TestUser1.
9. Verify that it is working by accessing a SIM_UNCLASSIFIED directory by typing “**http://192.168.0.130/twiki/bin/view/SIM_UNCLASSIFIED/Test_Integration /TestTopic”** as the URL.
10. Click **“Edit”**.
11. Type in some text, and click **“Save”**.
12. Expecting successful modification and display of the modified content.
13. Close the browser window.
14. Annotate completion by initialing box to the left.

2.9.2. Switch to WebDAV Test Instruction (D2)

1. Establish a SIM_UNCLASSIFIED session for mdemo1 user to wiki server using the TCBE, for the client machine.

2. In a new browser window connect to TWiki on the server. Type: "**http://192.168.0.130/twiki/bin/view/Main/WebHome"** as the URL.
3. Login to TWiki as TestUser1.
4. Verify that it is working by accessing a SIM_UNCLASSIFIED directory by typing "**http://192.168.0.130/twiki/bin/view/SIM_UNCLASSIFIED/Test_Integration /TestTopic"** as the URL.
5. Click **"Edit"**.
6. Type in some text, and click **"Save"**.
7. Expecting successful modification and display of the modified content.
8. Close the browser window.
9. Logout from the wiki server. Type: "logout" at the TCBE.
10. Establish a SIM_UNCLASSIFIED session for mdemo1 user to MYSEA MLS server using the TCBE, for the Client machine.
11. In a new explorer window connect to WebDAV on the MYSEA MLS server. Type: "**192.168.0.131\dav"** as the URL.
12. Verify that it is working by editing any file.
13. Close the explorer window.
14. Annotate completion by initialing box to the left.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C: MLS TWIKI USER GUIDE

This appendix provides a simple user guide for the MLS TWiki running on the MYSEA server. Detailed documentation of standard TWiki can be found in the TWiki website at "http://twiki.org/cgi-bin/view/TWiki04x01/TWikiReferenceManual".

A. MLS TWIKI WEB ORGANIZATION

There are three webs in the default MLS TWiki installation, namely the Main web, the TWiki web, and the Sandbox web. The Main web is the home page. It is also the web where all users' homepages are created when users register for TWiki account. The TWiki web is the web where the online documentations and other administration pages, e.g., user registration and password change, are stored. The Sandbox web is the place where users can create temporary topics for testing. These three webs are created at the SIM_UNCLASSIFIED level, i.e., to edit the web, users must be logged in at the SIM_UNCLASSIFIED level.

Besides the three default webs, the MLS TWiki has seven more webs, each created at a different security levels. Users will create contents in the appropriate web that corresponds to the security labels of the contents. The seven webs and their corresponding levels are:

- SIM_UNCLASSIFED web, sl1:il0
- SIM_CONFIDENTIAL web, sl3:il0
- SIM_SECRET web, sl5(sc1):il0
- SIM_PACIFIC_SECRET web, sl5(sc2): il0
- SIM_NATO_SECRET web, sl5(sc3):il0
- COALITION_COMMAND web, sl5(sc1 sc2 sc3): il0
- SIM_TOP_SECRET web, sl7:il0

B. GETTING STARTED

1. Log in the MLS wiki server through the TPE or TCBE software. Set session level to SIM_UNCLASSIFIED.

2. At the SIM_UNCLASSIFIED level, users will be able to register for new TWiki accounts, change password for existing TWiki accounts, and create test pages in Sandbox. Users are also able to read and edit contents within the Main, TWiki, and SIM_UNCLASSIFIED web in accordance to the permitted TWiki DAC settings of the respective contents. Users will not be able to read or edit contents from other webs.

3. To read or edit contents at other security level, set the session level to be equal to that security level. For example, to read or edit a page at SIM_SECRET, set the session level to SIM_SECRET using the TPE or TCBE software. Users will be able to read down, i.e. at SIM_SECRET, users will be able to read contents in the SIM_SECRET, SIM_CONFIDENTIAL and SIM_UNCLASSIFIED webs, as well as the default webs of Main, TWiki and Sandbox, in accordance to the TWiki DAC settings. However, the users are not able to read contents at security levels higher than SIM_SECRET. Users can only write at the level equal to their session level, i.e. the users can only edit contents in SIM_SECRET web, and only if TWiki DAC settings permit. At security levels other than SIM_UNCLASSIFIED, users are not able to register for new TWiki accounts or change password for existing TWiki accounts.

C. BASIC OPERATIONS

1. Registering for new account

- Login to TPE or TCBE using STOP user ID and password, at SIM_UNCLASSIFIED level.
- Open a browser, and type the following as the URL: "http://192.168.0.130/twiki/bin/view/Main/WebHome".
- For first time users, click the "register" link. Fill in the following mandatory fields with the appropriate values, and click "Submit":
    - First Name:

    - Last Name:
    - Email Address:
    - Your Password:
    - Retype password:
    - Country:
- The user will be prompted with a "Thank you for registering" message, and will be able to start using the TWiki system.

2. Editing a topic:

- Login to TPE or TCBE using STOP user ID and password, and select the appropriate session level, e.g., SIM_SECRET.
- Open a browser, and type the following as the URL: "http://192.168.0.130/twiki/bin/view/Main/WebHome".
- Login to TWiki using TWiki user ID and password.
- Browse to the appropriate web, e.g., SIM_SECRET web.
- Click **"Edit"** at the bottom of the topic.
- Enter the changes.
- Click **"Save"** to save the changes.

3. Creating a topic

- In an existing page, create a link to the new page by typing [[*newpage*]], substituting *newpage* with the appropriate name.
- Save the changes by clicking **"Save"**. User will then be presented with the content of the newly modified page.
- Click on the "?" beside the *newpage* text. User will be presented with the text edit form. Type in the content and click **"Save"**. The *newpage* topic will be created.

4. Deleting a topic

- Click on "More topic actions" on the lower right corner of the topic to be deleted.
- Click "**Delete topic...**, looking for references in *all public webs* **(recommended)**" option.
- Click "Delete" in the next screen.
- The topic will be deleted.

5. Edit/create a topic at other levels:

- Go to TPE or TCBE, type *sl* and type the security level to change to, e.g., to change to SIM_TOP_SECRET, type “SIM_TOP_SECRET”.
- Browser to the appropriate webs or topics and follow the procedure for editing, creating or deleting pages as outlined above.

LIST OF REFERENCES

[1]. Thuy D. Nguyen, Timothy E. Levin, T. E. and Cynthia E. Irvine, "MYSEA Testbed," Proceedings of the 6th IEEE Systems, Man and Cybernetics Information Assurance Workshop, West Point, NY, June 2005, pp. 438-439.

[2]. Peter H. Carstensen and Kjeld Schmidt, "Computer Supported Cooperative Work: New Challenges to Systems Design", In K. Itoh (Ed.), Handbook of human factors, 1999.

[3]. Mass Collaboration, Wikipedia, http://en.wikipedia.org/wiki/Mass_collaboration, dated 7 June 2007, last viewed 30 July 2007.

[4]. Wiki. Wikipedia, http://en.wikipedia.org/wiki/Wiki, dated 30 July 2007, last viewed 30 July 2007.

[5]. Lars Aronsson, "Operation of a Large Scale, General Purpose Wiki Website", Proceedings of the 6th International ICCC/IFIP Conference on Electronic Publishing, November 2002, pp 27-37.

[6]. Cynthia E. Irvine, Timothy E. Levin, Thuy D. Nguyen, David Shifflett, Jean Khosalim, Paul C. Clark, Albert Wong, Francis Afinidad, David Bibighaus, and Joseph Sears, "Overview of a High Assurance Architecture for Distributed Multilevel Security", Proceedings of the 2004 IEEE Systems, Man and Cybernetics Information Assurance Workshop, West Point, NY, June 2004, pp 38-45.

[7]. Jeremiah A. Bradney, "Use of WebDAV to Support a Virtual File System in a Coalition Environment", Master's Thesis, Naval Postgraduate School, Monterey, CA, June 2006.

[8]. David E. Bell and Leonard J. LaPadula, "Secure Computer System: Unified Exposition and Multics Interpretation", ESD-TR-75-306, Electronic Systems Division (AFSC) 1976.

[9]. K. J. Biba, "Integrity Considerations for Secure Computer Systems", MTR-3153, Mitre Corp., Bedford, M.A, April 1977.

[10]. 2007 MYSEA poster, http://cisr.nps.navy.mil/downloads/07poster_mysea.pdf, last viewed 1 November 2007.

[11]. Wiki engines, http://c2.com/cgi-bin/wiki?WikiEngines, last viewed 2 March 2007.

[12]. Wikimatrix. http://www.wikimatrix.org/, last viewed 2 March 2007.

[13]. Wiki Engines, http://c2.com/cgi-bin/wiki?WikiEngines, last viewed 2 March 2007.

[14]. Comparison of wiki software, http://en.wikipedia.org/wiki/Comparison_of_wiki_software, dated 1 March 2007, last viewed 2 March 2007.

[15]. Sonia Bui, "Single Sign-on Solution for MYSEA Services", Master's Thesis, Naval Postgraduate School, Monterey, CA, September 2005.

[16]. XTS-400, STOP 6.0, User's Manual, Document ID: XTDOC0005-01, Getronics Government Solutions, LLC, Herndon, VA, August 2002.

[17]. TWiki Access Control, http://twiki.org/cgi-bin/view/TWiki/TWikiAccessControl, dated 27 September 2007, last viewed 30 October 2007.

[18]. Dylan McNamee, Scott Heller and Dave Huff, "Building Multilevel Secure Web Services-Based Components for the Global Information Grid", STSC CrossTalk, May 2006.

[19]. Thuy D. Nguyen, private conversation at Naval Postgraduate School, 5 September 2007.

[20]. PmWiki, "http://www.pmwiki.org/wiki/PmWiki/PmWiki", dated 26 March 2006, last viewed 08 November 2007.

[21]. TWiki, "http://www.twiki.org/", last viewed 08 November 2007.

[22]. File System Permission, "http://en.wikipedia.org/wiki/Unix_permission", dated 22 November 2007, last viewed 26 November 2007.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Susan Alexander
   OASD/NII DOD/CIO
   Washington, DC

4. Hugo A. Badillo
   NSA
   Fort Meade, MD

5. George Bieber
   OSD
   Washington, DC

6. John Campbell
   National Security Agency
   Fort Meade, MD

7. Deborah Cooper
   DC Associates, LLC
   Roslyn, VA

8. Dr. Grace Crowder
   NSA
   Fort Meade, MD

9. Louise Davidson
   National Geospatial Agency
   Bethesda, MD

10. Steve Davis
    NRO
    Chantilly, VA

11. Vincent J. DiMaria
National Security Agency
Fort Meade, MD

12. Dr. Tim Fossum
National Science Foundation

13. Jennifer Guild
SPAWAR
Charleston, SC

14. Steve LaFountain
NSA
Fort Meade, MD

15. Dr. Greg Larson
IDA
Alexandria, VA

16. Dr. Karl Levitt
NSF
Arlington, VA

17. Dr. John Monastra
Aerospace Corporation
Chantilly, VA

18. John Mildner
SPAWAR
Charleston, SC

19. Mark T. Powell
Federal Aviation Administration
Washington, DC

20. Jim Roberts
Central Intelligence Agency
Reston, VA

21. Keith Jarren
NSA
Fort Meade, MD

22. Ed Schneider
IDA
Alexandria, VA

23. Mark Schneider
NSA
Fort Meade, MD

24. Keith Schwalm
Good Harbor Consulting, LLC
Washington, DC

25. Ken Shotting
NSA
Fort Meade, MD

26. CDR Wayne Slocum
SPAWAR
San Diego, CA

27. Dr. Ralph Wachter
ONR
Arlington, VA

28. Matt Warnock
Booze-Allen-Hamilton

29. Ed Bryant
Unified Cross Domain Management Office
MD

30. John P. Mcgeehan
Unified Cross Domain Management Office
MD

31. Paul A. Livingston
Unified Cross Domain Management Office
MD

32. Boyd Fletcher
SPAWAR
San Diego, CA

33. Mike Harrison
SPAWAR
San Diego, CA

34. Dr. Cynthia E. Irvine
Naval Postgraduate School
Monterey, CA

35. Thuy D. Nguyen
Naval Postgraduate School
Monterey, CA

36. Prof. Yeo Tat Soon
Director, Temasek Defense Systems Institute (TDSI)
National University of Singapore
Singapore

37. Tan Lai Poh
Temasek Defense Systems Institute (TDSI)
National University of Singapore
Singapore

38. Tan Sian Boon
Human Resource Department
Defense Science & Technology Agency
Singapore

39. Ong Kar Leong
Affiliation (SFS students: Civilian, Naval Postgraduate School)
Monterey, CA